aws

Guía para desarrolladores

Amazon API Gateway



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

Amazon API Gateway: Guía para desarrolladores

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

Las marcas comerciales y la imagen comercial de Amazon no se pueden utilizar en relación con ningún producto o servicio que no sea de Amazon de ninguna manera que pueda causar confusión entre los clientes y que menosprecie o desacredite a Amazon. Todas las demás marcas registradas que no son propiedad de Amazon son propiedad de sus respectivos propietarios, que pueden o no estar afiliados, conectados o patrocinados por Amazon.

Table of Contents

¿Qué es Amazon API Gateway? 1
Arquitectura de API Gateway 2
Características de API Gateway 3
Casos de uso de API Gateway 3
Uso de API Gateway para crear API REST 4
Uso de API Gateway para crear API HTTP 5
Uso de API Gateway para crear API de WebSocket 5
¿Quiénes utilizan API Gateway? 6
Acceso a API Gateway 7
Parte de la infraestructura sin servidor de AWS 8
Cómo comenzar a usar Amazon API Gateway 8
Conceptos de API Gateway 8
Elección entre las API de REST y las API de HTTP 15
Tipo de punto de conexión 15
Seguridad 15
Autorización 16
Administración de API 16
Desarrollo 17
Supervisión 18
Integraciones 18
Introducción a la consola de la API de REST 19
Paso 1: Crear una función Lambda 20
Paso 2: Crear una API de REST 21
Paso 3: Crear una integración de proxy de Lambda 21
Paso 4: Implementar la API 22
Paso 5: Invocar la API 22
(Opcional) Paso 6: limpiar 23
Requisitos previos 25
Cómo crear una Cuenta de AWS 25
Introducción 26
Paso 1: Crear una función Lambda 27
Paso 2: crear una API HTTP 27
Paso 3: probar la API 28
(Opcional) Paso 4: limpiar 29

Pasos siguientes 30
Tutoriales y talleres 32
Tutoriales sobre API REST 33
Elección de un tutorial de integración de Lambda 33
Tutorial: Crear una API de REST importando un ejemplo 58
Elección de un tutorial de integración de HTTP 68
Tutorial: Creación de una API de REST con una integración privada 82
Tutorial: Creación de una API de REST con una integración de AWS 88
Tutorial: Creación de una API de REST de calculadora con tres integraciones 94
Tutorial: Creación de una API de REST como proxy de Amazon S3 122
Tutorial: Creación de una API de REST como proxy de Amazon Kinesis 167
Tutorial: Creación de una API de REST mediante SDK de AWS o la AWS CLI 213
Tutorial: Creación de una API de REST privada 247
Tutoriales sobre API HTTP 253
Tutorial: Creación de una API de HTTP CRUD con Lambda y DynamoDB 253
Tutorial: Creación de una API de HTTP con una integración privada en Amazon ECS 266
Tutoriales de la API de WebSocket 273
Tutorial: Creación de una aplicación de chat de WebSocket 274
Tutorial: Creación de una API de WebSocket con una integración de AWS 279
API de REST de API Gateway 293
Desarrollo 293
Recurso de ejemplo para una API de REST 294
Características adicionales de la API de REST para el desarrollo 297
Tipos de puntos de conexión de API Gateway 299
Políticas de seguridad 303
Tipos de direcciones IP para API de REST en API Gateway 325
Métodos 327
Control de acceso 348
Integraciones 436
Validación de solicitudes 533
Transformaciones de datos 563
Respuestas de gateway 648
CORS 661
Tipos de medios binarios 676
Invocación 708
OpenAPI 743

Publicación ... 758
Implementación de las API de REST ... 758
Nombres de dominio personalizados ... 807
Optimizar ... 882
Configuración de caché ... 882
Codificación de contenido ... 897
Distribuir ... 903
Planes de uso ... 903
Documentación de la API ... 931
Generación de SDK ... 997
Vender sus API como SaaS ... 1024
Proteger ... 1029
Requerimiento de certificados de cliente ... 1029
Certificados de cliente de backend ... 1036
AWS WAF ... 1086
Limitación controlada ... 1089
API de REST privadas ... 1092
Supervisión ... 1152
Métricas de CloudWatch ... 1153
CloudWatch Logs ... 1162
X-Ray ... 1193
Portales de API Gateway ... 1208
Consideraciones ... 1208
Producto del portal ... 1209
Ejemplo de producto del portal de adopción de mascotas ... 1209
Siguientes pasos ... 1211
Creación de un producto del portal ... 1212
Creación de un punto de conexión REST del producto ... 1214
Creación de una página de producto ... 1216
Actualización de un producto del portal ... 1218
Actualización de un punto de conexión REST del producto ... 1219
Actualización de página de producto ... 1220
Uso compartido de productos de portal ... 1220
Habilitación de una prueba para un punto de conexión REST de un producto en el portal ... 1226
Eliminación de un producto del portal ... 1227
Creación de un portal ... 1228

Elección de un nombre de dominio ... 1228
Creación de un portal ... 1229
Actualización de un portal ... 1231
Actualización de un portal ... 1231
Vista previa de un portal ... 1232
Consideraciones ... 1232
Vista previa de un portal ... 1232
Publicación de un portal ... 1232
Consideraciones ... 1233
Publicación de un portal ... 1233
Uso de un portal ... 1233
Uso de un portal ... 1233
Solución de problemas del portal ... 1234
Desactivación de un portal ... 1235
Desactivación de un portal ... 1235
Eliminación de un portal ... 1236
Eliminación de un portal ... 1236
API de HTTP de API Gateway ... 1237
Desarrollo ... 1237
Creación de una API HTTP ... 1238
Rutas ... 1239
Tipos de direcciones IP para las API HTTP en API Gateway ... 1242
Control de acceso ... 1244
Integraciones ... 1264
CORS ... 1283
Asignación de parámetros ... 1286
OpenAPI ... 1293
Publicación ... 1303
Etapas ... 1304
Política de seguridad de las API de HTTP en API Gateway ... 1306
Nombres de dominio personalizados ... 1308
Proteger ... 1318
Limitación ... 1318
TLS mutua ... 1320
Supervisión ... 1327
Métricas ... 1327

Registro ... 1330
Resolución de problemas ... 1341
Integraciones de Lambda ... 1341
Autorizadores de JWT ... 1344
API de WebSocket de API Gateway ... 1346
Descripción general de las API de WebSocket ... 1346
Uso de rutas para una API de WebSocket ... 1347
Envío de datos a las aplicaciones cliente conectadas ... 1348
Códigos de estado de la API de WebSocket ... 1348
Administración de usuarios conectados y aplicaciones cliente ... 1349
Invocación de la integración del backend ... 1352
Expresiones de selección de WebSocket ... 1357
Desarrollo ... 1368
Crear y configurar ... 1368
Tipos de direcciones IP para las API de WebSocket en API Gateway ... 1370
Rutas ... 1372
Control de acceso ... 1380
Integraciones ... 1389
Validación de solicitudes ... 1399
Transformaciones de datos ... 1402
Tipos de medios binarios ... 1416
Invocación ... 1416
Publicación ... 1420
Etapas ... 1421
Implementación de una API de WebSocket ... 1423
Política de seguridad de las API de WebSocket en API Gateway ... 1426
Nombres de dominio personalizados ... 1428
Protección ... 1436
Limitación de nivel de cuenta por región ... 1436
Limitación de nivel de ruta ... 1437
Supervisar ... 1437
Métricas ... 1438
Registro ... 1440
ARN de API Gateway ... 1449
Recursos de API HTTP y API de WebSocket ... 1449
Recursos de API de REST ... 1452

Recursos de nombres de dominio 1456
execute-api (API HTTP, API de WebSocket y API de REST) 1457
Extensiones de OpenAPI 1459
x-amazon-apigateway-any-method 1460
Ejemplos de x-amazon-apigateway-any-method 1461
x-amazon-apigateway-cors 1462
Ejemplo de x-amazon-apigateway-cors 1462
x-amazon-apigateway-api-key-source 1463
Ejemplo de x-amazon-apigateway-api-key-source 1464
x-amazon-apigateway-auth 1465
Ejemplo de x-amazon-apigateway-auth 1465
x-amazon-apigateway-authorizer 1466
Ejemplos de x-amazon-apigateway-authorizer para API REST 1469
Ejemplos de x-amazon-apigateway-authorizer para API HTTP 1473
x-amazon-apigateway-authtype 1475
Ejemplo de x-amazon-apigateway-authtype 1475
Véase también 1478
x-amazon-apigateway-binary-media-type 1478
Ejemplo de x-amazon-apigateway-binary-media-types 1478
x-amazon-apigateway-documentation 1478
Ejemplo de x-amazon-apigateway-documentation 1478
x-amazon-apigateway-endpoint-access-mode 1479
x-amazon-apigateway-endpoint-access-modeEjemplo de 1479
x-amazon-apigateway-endpoint-configuration 1479
Ejemplos de x-amazon-apigateway-endpoint-configuration 1481
x-amazon-apigateway-gateway-responses 1481
Ejemplo de x-amazon-apigateway-gateway-responses 1482
x-amazon-apigateway-gateway-responses.gatewayResponse 1482
Ejemplo de x-amazon-apigateway-gateway-responses.gatewayResponse 1483
x-amazon-apigateway-gateway-responses.responseParameters 1483
Ejemplo de x-amazon-apigateway-gateway-responses.responseParameters 1484
x-amazon-apigateway-gateway-responses.responseTemplates 1484
Ejemplo de x-amazon-apigateway-gateway-responses.responseTemplates 1485
x-amazon-apigateway-importexport-version 1485
Ejemplo de x-amazon-apigateway-importexport-version 1486
x-amazon-apigateway-integration 1486

Ejemplos de x-amazon-apigateway-integration 1494
x-amazon-apigateway-integrations 1496
Ejemplo de x-amazon-apigateway-integrations 1496
x-amazon-apigateway-integration.requestTemplates 1498
Ejemplo de x-amazon-apigateway-integration.requestTemplates 1498
x-amazon-apigateway-integration.requestParameters 1499
`x-amazon-apigateway-integration.requestParameters`Ejemplo de 1500
x-amazon-apigateway-integration.responses 1501
`x-amazon-apigateway-integration.responses`Ejemplo de 1502
x-amazon-apigateway-integration.response 1503
`x-amazon-apigateway-integration.response`Ejemplo de 1504
x-amazon-apigateway-integration.responseTemplates 1505
Ejemplo de x-amazon-apigateway-integration.responseTemplate 1505
x-amazon-apigateway-integration.responseParameters 1506
`x-amazon-apigateway-integration.responseParameters`Ejemplo de 1506
x-amazon-apigateway-integration.tlsConfig 1506
Ejemplos de x-amazon-apigateway-integration.tlsConfig 1509
x-amazon-apigateway-minimum-compression-size 1510
Ejemplo de x-amazon-apigateway-minimum-compression-size 1510
x-amazon-apigateway-policy 1510
`x-amazon-apigateway-policy`Ejemplo de 1510
x-amazon-apigateway-request-validator 1511
`x-amazon-apigateway-request-validator`Ejemplo de 1512
x-amazon-apigateway-request-validators 1512
`x-amazon-apigateway-request-validators`Ejemplo de 1513
x-amazon-apigateway-request-validators.requestValidator 1514
`x-amazon-apigateway-request-validators.requestValidator`Ejemplo de 1514
x-amazon-apigateway-security-policy 1514
`x-amazon-apigateway-security-policy`Ejemplo de 1515
x-amazon-apigateway-tag-value 1515
`x-amazon-apigateway-tag-value`Ejemplo de 1515
Seguridad 1517
Protección de los datos 1518
Cifrado de datos 1519
Privacidad del tráfico entre redes 1521
Administración de identidades y accesos 1521

Público ... 1522
Autenticación con identidades ... 1522
Administración del acceso con políticas ... 1523
Cómo funciona Amazon API Gateway con IAM ... 1525
Ejemplos de políticas basadas en identidades ... 1530
Ejemplos de políticas basadas en recursos ... 1543
Solución de problemas ... 1543
Uso de roles vinculados a servicios ... 1545
Registro y monitorización ... 1550
Trabajar con CloudTrail ... 1552
Trabajo con AWS Config ... 1555
Validación de conformidad ... 1559
Resiliencia ... 1559
Seguridad de la infraestructura ... 1560
Configuración y análisis de vulnerabilidades ... 1561
Prácticas recomendadas ... 1561
Etiquetado ... 1563
Recursos de API Gateway que se pueden etiquetar ... 1564
Herencia de etiquetas en la API V1 de Amazon API Gateway ... 1565
Restricciones de etiquetas y convenciones de uso ... 1566
Control de acceso basado en atributos ... 1567
Limitar acciones en función de etiquetas de recursos ... 1567
Permitir acciones en función de las etiquetas de recursos ... 1568
Denegar operaciones de etiquetado ... 1569
Permitir operaciones de etiquetado ... 1570
Referencias de API ... 1572
Puntos de conexión del servicio de API Gateway ... 1572
Puntos de conexión IPv4 ... 1572
Puntos de conexión de pila doble (IPv4 e IPv6) ... 1573
Especificación de puntos de conexión ... 1573
Uso de direcciones IPv6 en políticas de IAM ... 1574
Cuotas y notas importantes ... 1576
Cuotas de nivel de cuenta de API Gateway, por región ... 1576
Cuotas de API Gateway para crear, implementar y administrar una API ... 1577
Cuotas de API de REST ... 1581
Prácticas recomendadas para reducir las cuotas ... 1588

Cuotas de API HTTP 1589
Prácticas recomendadas para reducir las cuotas 1593
Cuotas del portal 1593
Cuotas de API de WebSocket 1594
Notas importantes 1597
Notas importantes de Amazon API Gateway para las API de HTTP 1598
Notas importantes de Amazon API Gateway para las API HTTP y de WebSocket 1598
Notas importantes para las API de REST y las API de WebSocket 1598
Notas importantes para las API WebSocket 1599
Notas importantes para las API de REST 1599
Historial de revisión 1606
Actualizaciones anteriores 1623

¿Qué es Amazon API Gateway?

Amazon API Gateway es un servicio de AWS para la creación, la publicación, el mantenimiento, el monitoreo y la protección de las API REST, HTTP y de WebSocket a cualquier escala. Los desarrolladores de API pueden crear API que obtengan acceso a AWS o a otros servicios web, así como los datos almacenados en la nube de AWS. Como desarrollador de API de API Gateway, puede crear API para su uso en sus propias aplicaciones de cliente. También puede ofrecer sus API a otros desarrolladores de aplicaciones externos. Para obtener más información, consulte the section called “¿Quiénes utilizan API Gateway?”.

API Gateway crea API RESTful que:

- Se basan en HTTP.
- Habilitan la comunicación entre cliente y servidor sin estado.
- Implementan métodos HTTP estándar como, por ejemplo, GET, POST, PUT, PATCH y DELETE.

Para obtener más información acerca de las API REST de API Gateway y las API HTTP, consulte the section called “Elección entre las API de REST y las API de HTTP ”, API de HTTP de API Gateway, the section called “Uso de API Gateway para crear API REST” y the section called “Desarrollo”.

API Gateway crea API de WebSocket que:

- Cumplen el protocolo WebSocket, que permite la comunicación entre el cliente y el servidor de dúplex completo con estado.
- Dirigen mensajes entrantes en función del contenido de los mensajes.

Para obtener más información sobre las API de WebSocket de API Gateway, consulte the section called “Uso de API Gateway para crear API de WebSocket” y the section called “Descripción general de las API de WebSocket”.

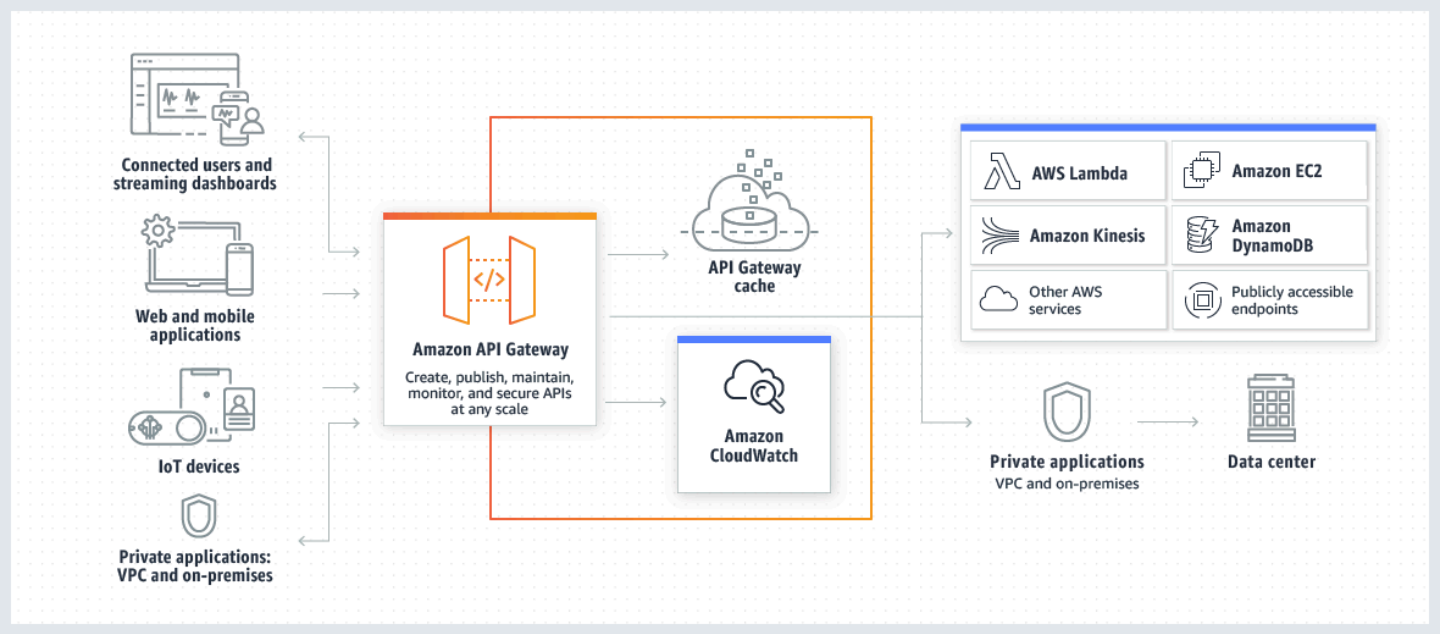
Temas

- Arquitectura de API Gateway
- Características de API Gateway
- Casos de uso de API Gateway
- Acceso a API Gateway

- Parte de la infraestructura sin servidor de AWS
- Cómo comenzar a usar Amazon API Gateway
- Conceptos de Amazon API Gateway
- Elección entre las API de REST y las API de HTTP
- Introducción a la consola de la API de REST

Arquitectura de API Gateway

En el siguiente diagrama se muestra la arquitectura de API Gateway.



Este diagrama ilustra cómo las API que crea en Amazon API Gateway le proporcionan a usted o a sus clientes desarrolladores una experiencia de desarrollador integrada y coherente para crear aplicaciones sin servidor de AWS. API Gateway gestiona todas las tareas relacionadas con la aceptación y el procesamiento de centenares de miles de llamadas simultáneas a la API. Estas tareas incluyen la administración del tráfico, el control de la autorización y el acceso, el monitoreo y la administración de versiones de la API.

API Gateway actúa como una "puerta principal" para que las aplicaciones accedan a datos, lógica empresarial o funcionalidad desde sus servicios de backend, como cargas de trabajo que se ejecutan en Amazon Elastic Compute Cloud (Amazon EC2), código que se ejecuta en AWS Lambda, cualquier aplicación web o aplicaciones de comunicación en tiempo real.

Características de API Gateway

Amazon API Gateway ofrece características como las siguientes:

- Compatibilidad con las API con estado (WebSocket) y las API (HTTP y REST).
- Mecanismos de autenticación eficaces y flexibles, como políticas de AWS Identity and Access Management, funciones de autorizador de Lambda y grupos de usuarios de Amazon Cognito.
- Implementaciones de la versión Canary para el despliegue de cambios de forma segura.
- Registro de CloudTrail y monitoreo del uso y de los cambios en las API.
- Registro de acceso y registro de ejecución de CloudWatch, que incluye la posibilidad de establecer alarmas. Para obtener más información, consulte the section called “Métricas de CloudWatch” y the section called “Métricas”.
- Posibilidad de utilizar plantillas de CloudFormation para habilitar la creación de las API. Para obtener más información, consulte Referencia de tipos de recursos de Amazon API Gateway y Referencia de tipos de recursos de Amazon API Gateway V2.
- Soporte para los nombres de dominio personalizados.
- Integración con AWS WAF para la protección de sus API frente a ataques web comunes.
- Integración con AWS X-Ray para comprender y cribar latencias de rendimiento.

Para obtener una lista completa de lanzamientos de características de API Gateway, consulte Historial de revisión.

Casos de uso de API Gateway

En la siguiente sección de casos de uso se presenta una descripción general de los distintos tipos de API de API Gateway y de los distintos tipos de desarrolladores que utilizan API Gateway. Para obtener información más detallada sobre las diferencias entre las API HTTP y las API de REST, consulte the section called “Elección entre las API de REST y las API de HTTP ”.

Temas

- Uso de API Gateway para crear API REST
- Uso de API Gateway para crear API HTTP
- Uso de API Gateway para crear API de WebSocket

- ¿Quiénes utilizan API Gateway?

Uso de API Gateway para crear API REST

Una API REST de API Gateway está formada por recursos y métodos. Un recurso es una entidad lógica a la que una aplicación puede acceder a través de una ruta de recursos. Un método se corresponde con una solicitud de la API de REST enviada por el usuario de la API y la respuesta devuelta al usuario.

Por ejemplo, `/incomes` podría ser la ruta de un recurso que representa los ingresos del usuario de la aplicación. Un recurso puede tener una o varias operaciones, las cuales definen los verbos HTTP correspondientes, por ejemplo, GET, POST PUT, PATCH y DELETE. Una combinación de una ruta de recurso y una operación identifica un método de la API. Por ejemplo, un método `POST /incomes` podría añadir una ganancia devengada por el intermediario y un método `GET /expenses` podría consultar los gastos en los que ha incurrido el intermediario.

La aplicación no necesita saber dónde se almacenan y de dónde se obtienen los datos solicitados en el backend. En las API REST de API Gateway, el frontend se encapsula mediante solicitudes de métodos y respuestas de métodos. La API se conecta con el backend por medio de solicitudes de integración y respuestas de integración.

Por ejemplo, con DynamoDB como backend, el desarrollador de la API configura la solicitud de integración para reenviar la solicitud del método de entrada al backend elegido. La configuración incluye las especificaciones de acción de DynamoDB adecuadas y los roles y políticas de IAM necesarios, así como la transformación de los datos de entrada correcta. El backend devuelve el resultado a API Gateway como una respuesta de integración.

Para dirigir la respuesta de integración a una respuesta de método adecuada (de un determinado código de estado HTTP) al cliente, puede configurar la respuesta de integración para que mapee los parámetros de respuesta necesarios desde la integración del método. A continuación, si fuera necesario, convierta el formato de los datos de salida del backend al del frontend. API Gateway le permite definir un esquema o modelo de carga para facilitar la configuración de la plantilla de mapeo de cuerpo.

API Gateway proporciona funcionalidad de administración de la API REST, como a siguiente:

- Soporte para generar SDK y crear la documentación de API mediante extensiones de OpenAPI de API Gateway

- Limitación controlada de solicitudes HTTP

Uso de API Gateway para crear API HTTP

Las API HTTP le permiten crear API de RESTful con menor latencia y menor costo que las API de REST.

Puede utilizar las API HTTP para enviar solicitudes a funciones de AWS Lambda o a cualquier punto de conexión HTTP públicamente direccionable.

Por ejemplo, puede crear una API HTTP que se integre con una función de Lambda en el backend. Cuando un cliente llama a la API, API Gateway envía la solicitud a la función de Lambda y devuelve la respuesta de la función al cliente.

Las API HTTP son compatibles con la autorización de OpenID Connect y OAuth 2.0. Vienen con soporte integrado para el uso compartido de recursos entre orígenes (CORS) y las implementaciones automáticas.

Para obtener más información, consulte the section called “Elección entre las API de REST y las API de HTTP ”.

Uso de API Gateway para crear API de WebSocket

En una API de WebSocket, el cliente y el servidor pueden enviarse mensajes entre sí en cualquier momento. Los servidores de backend pueden enviar fácilmente datos a los usuarios y dispositivos conectados, lo que evita la necesidad de implementar complejos mecanismos de sondeo.

Por ejemplo, podría crear una aplicación sin servidor mediante una API de WebSocket de API Gateway y AWS Lambda para enviar y recibir mensajes entre usuarios individuales o grupos de usuarios de una sala de conversación. También puede invocar servicios de backend como por ejemplo AWS Lambda, Amazon Kinesis o un punto de conexión HTTP en función del contenido de los mensajes.

Puede utilizar la API de WebSocket de API Gateway para crear aplicaciones seguras de comunicación en tiempo real sin tener que aprovisionar ni administrar servidores para administrar las conexiones o los intercambios de datos a gran escala. Los casos de uso focalizados incluyen aplicaciones en tiempo real, como la siguiente:

- Aplicaciones de chat

- Paneles en tiempo real como los de cotizaciones bursátiles
- Alertas y notificaciones en tiempo real

API Gateway proporciona funcionalidad de administración de la API de WebSocket, como la siguiente:

- Monitoreo y limitación controlada de las conexiones y los mensajes
- Uso de AWS X-Ray para rastrear los mensajes que se transfieren a través de las API a los servicios de backend
- Integración sencilla con puntos de conexión HTTP/HTTPS

¿Quiénes utilizan API Gateway?

Existen dos tipos de desarrolladores que utilizan API Gateway: los desarrolladores de API y los desarrolladores de aplicaciones.

Un desarrollador de API crea e implementa una API para habilitar la funcionalidad necesaria en API Gateway. El desarrollador de API debe ser un usuario en la cuenta de AWS que posee la API.

Un desarrollador de aplicaciones diseña una aplicación funcional que llama a los servicios de AWS mediante la invocación de una API de WebSocket o REST creada por un desarrollador de API en API Gateway.

El desarrollador de aplicaciones es el cliente del desarrollador de API. El desarrollador de aplicaciones no necesita tener una cuenta de AWS, siempre que la API no requiera permisos de IAM, o bien admita la autorización de usuarios a través proveedores de identidad federada de terceros admitidos por la Federación de identidades de grupos de usuarios de Amazon Cognito. Los proveedores de identidad incluyen a Amazon, el grupo de usuarios de Amazon Cognito, Facebook y Google.

Creación y administración de una API de API Gateway

Para crear, configurar e implementar una API, un desarrollador de API trabaja con el componente del servicio de API Gateway para administración de la API, denominado `apigateway`.

Como desarrollador de API, puede crear y administrar una API mediante la consola de API Gateway tal y como se describe en Introducción a API Gateway o llamando a la Referencias de API. Existen

varias maneras de llamar a esta API. Incluyen el uso de AWS Command Line Interface (AWS CLI), o mediante un SDK de AWS. Además, puede habilitar la creación de la API con Plantillas de AWS CloudFormation o, en el caso de las API REST y HTTP, con Extensiones de OpenAPI para API Gateway.

Para obtener una lista de las regiones en las que API Gateway está disponible, así como los puntos de conexión del servicio de control asociados, consulte Cuotas y puntos de conexión de Amazon API Gateway.

Llamadas a un API de API Gateway

Un desarrollador de aplicaciones trabaja con el componente de servicio de API Gateway para ejecución de la API, denominado `execute-api`, para invocar una API que se ha creado o implementado en API Gateway. La API creada expone las entidades de programación subyacentes. Existen varias maneras de llamar a dicha API. Para obtener más información, consulte Invocación de las API de REST en API Gateway y Invocación de las API de WebSocket.

Acceso a API Gateway

Puede obtener acceso a Amazon API Gateway de las siguientes formas:

- Consola de administración de AWS: la Consola de administración de AWS proporciona una interfaz web que permite crear y administrar las API. Una vez completados los pasos que se indican en Requisitos previos, puede acceder a la consola de API Gateway en https://console.aws.amazon.com/apigateway.
- AWS SDK: si utiliza un lenguaje de programación para el que AWS proporciona un SDK, puede usar un SDK para obtener acceso a API Gateway. Los SDK simplifican la autenticación, se integran fácilmente con su entorno de desarrollo y proporcionan acceso a los comandos de API Gateway. Para obtener más información, consulte Herramientas para Amazon Web Services.
- API de API Gateway V1 y V2: si utiliza un lenguaje de programación para el que no haya un SDK disponible, consulte la Referencia de la API de Amazon API Gateway versión 1 y la Referencia de la API de Amazon API Gateway versión 2.
- AWS Command Line Interface: para obtener más información, consulte Configuración inicial de la AWS Command Line Interface en la Guía del usuario de AWS Command Line Interface.
- AWS Tools for Windows PowerShell: para obtener más información, consulte Configuración de AWS Tools for Windows PowerShell en la Guía del usuario de Herramientas de AWS para PowerShell.

Parte de la infraestructura sin servidor de AWS

Junto con AWS Lambda, API Gateway es la parte de la infraestructura sin servidor de AWS orientada a la aplicación. Para obtener más información sobre cómo empezar a usar la tecnología sin servidor, consulte Guía para desarrolladores de tecnología sin servidor.

Para que una aplicación llame a los servicios de AWS disponibles públicamente, puede utilizar Lambda para interactuar con los servicios necesarios y exponer las funciones de Lambda a través de los métodos de API de API Gateway. AWS Lambda ejecuta el código en una infraestructura informática de alta disponibilidad. Realiza todos los procesos de ejecución y administración que necesitan los recursos informáticos. Para habilitar las aplicaciones sin servidor, API Gateway es compatible con las integraciones de proxy optimizadas con puntos de conexión de AWS Lambda y HTTP.

Cómo comenzar a usar Amazon API Gateway

Para obtener una introducción a Amazon API Gateway, consulte lo siguiente:

- Introducción, que proporciona una explicación para crear una API HTTP.
- Serverless land, que proporciona videos instructivos.
- Happy Little API Shorts, que es una serie de breves vídeos instructivos.

Conceptos de Amazon API Gateway

En la siguiente sección se describen los conceptos introductorios sobre el uso de API Gateway.

API Gateway

API Gateway es un servicio de AWS que admite lo siguiente:

- Crear, implementar y administrar una interfaz de programación de aplicaciones (API) RESTful para exponer los puntos de enlace HTTP del backend, funciones de AWS Lambda u otros servicios de AWS.
- Crear, implementar y administrar una API de WebSocket para exponer funciones de AWS Lambda u otros servicios de AWS.
- Invocar los métodos de API expuestos a través de los puntos de enlace HTTP y WebSocket del frontend.

API REST de API Gateway

Una colección de recursos y métodos HTTP que se integran con puntos de enlace HTTP de backend, funciones de Lambda u otros servicios de AWS. Puede implementar esta colección en una o más etapas. Normalmente, los recursos de la API están organizados en un árbol de recursos de acuerdo con la lógica de la aplicación. Cada recurso de la API puede exponer uno o varios métodos de la API que tienen verbos HTTP únicos admitidos por API Gateway. Para obtener más información, consulte the section called “Elección entre las API de REST y las API de HTTP ”.

API HTTP de API Gateway

Colección de rutas y métodos integrados con los puntos de enlace HTTP de backend o funciones de Lambda. Puede implementar esta colección en una o más etapas. Cada ruta puede exponer uno o varios métodos de la API que tienen verbos HTTP únicos admitidos por API Gateway. Para obtener más información, consulte the section called “Elección entre las API de REST y las API de HTTP ”.

API de WebSocket de API Gateway

Una colección de rutas y claves de ruta de WebSocket que se integran en puntos de enlace HTTP de backend, funciones de Lambda u otros servicios de AWS. Puede implementar esta colección en una o más etapas. Los métodos de la API se invocan a través de conexiones WebSocket del frontend que se pueden asociar a un nombre de dominio personalizado registrado.

Implementación de API

Una instantánea de un momento dado de la API de API Gateway. Para que los clientes puedan utilizar la implementación, esta debe asociarse a una o más etapas de API.

Desarrollador de la API

Su cuenta de AWS que es propietaria de una implementación de API Gateway (por ejemplo, un proveedor de servicios que también admite el acceso mediante programación).

Punto de enlace de la API

Un nombre de host para una API de API Gateway que se implementa en una región específica. El nombre de host tiene el formato `{api-id}.execute-api.{region}.amazonaws.com`. Se admiten los siguientes tipos de puntos de enlace de API:

- Punto de enlace de API optimizada para límites
- Punto de enlace de API privada

- Punto de enlace de API regional

Clave de API

Cadena alfanumérica que API Gateway utiliza para identificar a un desarrollador de aplicaciones que utiliza su API REST o de WebSocket. API Gateway puede generar claves de API en su nombre o usted puede importarlas desde un archivo CSV. Puede utilizar claves de API junto con autorizadores de Lambda o planes de uso para controlar el acceso a sus API.

Consulte los puntos de enlace de API.

Propietario de API

Consulte Desarrollador de la API.

Etapa de API

Una referencia lógica a un estado del ciclo de vida de la API (por ejemplo, 'dev', 'prod', 'beta', 'v2'). Las etapas de API se identifican por un ID y un nombre de etapa de API.

Desarrolladores de aplicaciones

Un creador de aplicaciones que puede tener o no una cuenta de AWS y que interactúa con la API que usted, el desarrollador de API, ha implementado. Los desarrolladores de aplicaciones son sus clientes. Un desarrollador de aplicaciones suele identificarse por una clave de API.

URL de devolución de llamada

Cuando un cliente nuevo se conecta a través de una conexión WebSocket, puede llamar a una integración en API Gateway para almacenar la URL de devolución de llamada del cliente. Esa URL de devolución de llamada se puede utilizar para enviar mensajes al cliente desde el sistema backend.

Portal para desarrolladores

Una aplicación en la que los proveedores de API pueden compartir sus API y su documentación con los consumidores de API. Las API se agrupan en productos, que son una recopilación de puntos de conexión de las API de REST, documentación de la API y documentación complementaria del producto.

Consulte los portales de API Gateway.

Punto de enlace de API optimizada para límites

El nombre de host predeterminado de una API de API Gateway implementada en la región especificada, mientras se utiliza una distribución de CloudFront para facilitar el acceso de los

clientes, normalmente, desde otras regiones de AWS. Las solicitudes de API se dirigen al punto de presencia de CloudFront más cercano, que normalmente mejora el tiempo de conexión para clientes en distintas ubicaciones geográficas.

Consulte los puntos de enlace de API.

Solicitud de integración

La interfaz interna de una ruta de API de WebSocket o un método de API REST en API Gateway, en la que se asignará el cuerpo de una solicitud de ruta o los parámetros y el cuerpo de una solicitud de método a los formatos requeridos por el backend.

Respuesta de integración

La interfaz interna de una ruta de API de WebSocket o un método de API REST en API Gateway, en la que se asignarán los códigos de estado, los encabezados y la carga que se reciben del backend al formato de respuesta que se devuelve a una aplicación cliente.

Plantilla de asignación

Un script escrito en Velocity Template Language (VTL), que transforma un cuerpo de la solicitud del formato de datos del frontend al formato de datos del backend o viceversa (en el caso del cuerpo de la respuesta). Las plantillas de asignación se pueden especificar en la solicitud de integración o en la respuesta de integración. Pueden hacer referencia a datos disponibles en tiempo de ejecución como contexto y variables de etapa.

La asignación puede ser tan sencilla como una transformación de identidad que transfiere los encabezados o el cuerpo a través de la integración tal cual desde el cliente al backend para una solicitud. Lo mismo sucede con la respuesta, en la que la carga se transmite desde el backend al cliente.

Solicitud de método

La interfaz pública de un método de una API de API Gateway que define los parámetros y el cuerpo que un desarrollador de aplicaciones debe enviar en las solicitudes para acceder al backend a través de la API.

Respuesta de método

La interfaz pública de una API de REST que define los códigos de estado, los encabezados y los modelos de cuerpo que un desarrollador de aplicaciones debería esperar en las respuestas de la API.

Integración simulada

En una integración simulada, las respuestas de la API se generan directamente a partir de API Gateway, sin necesidad de un backend de integración. Como desarrollador de una API, puede decidir cómo API Gateway responde a una solicitud de integración simulada. Para ello, configura la solicitud de integración y la respuesta de integración del método para asociar una respuesta a un código de estado determinado.

Modelo

Un esquema de datos que especifica la estructura de datos de una solicitud o carga de respuesta. Es necesario un modelo para generar un SDK con establecimiento inflexible de tipos de una API. También se utiliza para validar cargas. Un modelo es cómodo para generar una plantilla de asignación de muestra para iniciar la creación de una plantilla de asignación de producción. Aunque es útil, no se requiere para crear una plantilla de asignación.

Portal

Consulte el portal para desarrolladores.

Producto del portal

Un servicio o una funcionalidad que desea compartir. El producto del portal es una recopilación de puntos de conexión REST de productos y páginas de productos. Los puntos de conexión REST de productos son los puntos de acceso al producto del portal y constan de la ruta y el método de una API de REST y la fase en la que se implementa. Las páginas de productos son documentación que proporciona para explicar cómo los consumidores de API pueden utilizar los puntos de conexión de los productos. Puede compartir productos entre cuentas de AWS para agregarlos a los portales.

Consulte Creación de un producto.

API privado

Consulte Punto de enlace de API privada

Punto de enlace de API privada

Un punto de enlace de API que se expone a través de puntos de enlace de la VPC de tipo interfaz y permite que un cliente obtenga acceso seguro a los recursos de una API privada dentro de una VPC. Las API privadas están aisladas de la red pública de Internet y solo se puede obtener acceso a ellas a través de los puntos de enlace de la VPC de API Gateway a los que se les ha concedido acceso.

Integración privada

Tipo de integración de API Gateway que permite que un cliente tenga acceso a los recursos que se encuentran en la VPC de un usuario a través de un punto de enlace de una API REST privada sin exponerlos a la red pública de Internet.

Integración de proxy

Una configuración simplificada de integración de API Gateway. Puede configurar una integración de proxy como una integración de proxy HTTP o como una integración de proxy de Lambda.

En el caso de la integración de proxy HTTP, API Gateway transmite toda la solicitud y la respuesta entre el frontend y un backend HTTP. Para la integración de proxy Lambda, API Gateway envía toda la solicitud como entrada a una función de Lambda de backend. A continuación, API Gateway transforma el resultado de la función de Lambda en una respuesta HTTP del frontend.

En las API de REST, la integración de proxy se utiliza normalmente con un recurso de proxy, que se representa mediante una variable de ruta expansiva (por ejemplo, `{proxy+}`) combinada con un método catch-all `ANY`.

Creación rápida

Puede utilizar la creación rápida para simplificar la creación de una API HTTP. La creación rápida crea una API con una integración de Lambda o HTTP, una ruta de método catch-all predeterminada y una etapa predeterminada configurada para implementar automáticamente los cambios. Para obtener más información, consulte the section called “Crear una API HTTP mediante la AWS CLI”.

Punto de enlace de API regional

El nombre de host de una API implementada en la región especificada y pensada para prestar servicio a clientes, como, por ejemplo, instancias EC2, de la misma región de AWS. Las solicitudes de API se dirigen directamente a la API de API Gateway específica de la región sin pasar por ninguna distribución de CloudFront. Para las solicitudes destinadas a la misma región, existe un punto de enlace regional que evita el trayecto innecesario de ida y vuelta a una distribución de CloudFront.

Además, puede aplicar el direccionamiento basado en latencia a los puntos de enlace regionales para implementar una API en varias regiones con la misma configuración de punto de enlace de API regional, establecer el mismo nombre de dominio personalizado para cada API implementada

y configurar registros de DNS basados en latencia en Route 53 para dirigir las solicitudes de los clientes a la región que tiene la mínima latencia.

Consulte los puntos de enlace de API.

Ruta

Se utiliza una ruta de WebSocket en API Gateway para dirigir los mensajes entrantes a una integración específica, como una función de AWS Lambda, según el contenido de los mensajes. Al definir la API de WebSocket, se especifica un clave de ruta y un backend de integración. La clave de ruta es un atributo del cuerpo del mensaje. Cuando se encuentra una coincidencia para la clave de ruta en un mensaje entrante, se invoca el backend de integración.

También se puede establecer una ruta predeterminada para las claves de ruta no coincidentes o para especificar un modelo de proxy que transfiera el mensaje tal cual a los componentes del backend que realizan el direccionamiento y procesan la solicitud.

Solicitud de ruta

La interfaz pública de un método de API de WebSocket en API Gateway que define el cuerpo que un desarrollador de aplicaciones debe enviar en las solicitudes para obtener acceso al backend a través de la API.

Respuesta de ruta

La interfaz pública de una API de WebSocket que define los códigos de estado, los encabezados y los modelos de cuerpo que un desarrollador de aplicaciones debería esperar de API Gateway.

Plan de uso

Un plan de uso ofrece a determinados clientes de la API acceso a una o varias API de WebSocket o de REST implementadas. Puede utilizar un plan de uso para configurar la limitación controlada y los límites de cuota que se ejecutarán en claves de API de cliente individual.

Conexión WebSocket

API Gateway mantiene una conexión persistente entre los clientes y API Gateway. No hay conexión persistente entre API Gateway y las integraciones de backend, como las funciones de Lambda. Los servicios de backend se invocan según sea necesario, en función del contenido de los mensajes recibidos de los clientes.

Elección entre las API de REST y las API de HTTP

Las API de REST y las API HTTP son productos API de RESTful. Las API de REST admiten más funciones que las API HTTP, mientras que las API HTTP están diseñadas con características mínimas para que puedan ofrecerse a un precio más bajo. Elija las API de REST si necesita características como claves de API, limitación por cliente, validación de solicitudes, integración de AWS WAF o puntos de conexión de API privados. Elija las API de HTTP si no necesita las funciones incluidas con las API de REST.

En las siguientes secciones se resumen las características principales disponibles en las API de REST y las API HTTP. Cuando es necesario, se proporcionan enlaces adicionales para navegar entre las secciones de la API de REST y la API de HTTP de la Guía para desarrolladores de API Gateway.

Tipo de punto de conexión

El tipo de punto de conexión hace referencia al punto de conexión que API Gateway crea para su API. Para obtener más información, consulte the section called “Tipos de puntos de conexión de API Gateway”.

Tipo de punto de conexión	API de REST	API HTTP
Optimizada para la periferia	Sí	No
Regional	Sí	Sí
Private	Sí	No

Seguridad

API Gateway proporciona una serie de formas de proteger su API de ciertas amenazas, como actores malintencionados o picos de tráfico. Para obtener más información, consulte the section called “Proteger” y the section called “Proteger”.

Características de seguridad	API de REST	API HTTP
Autenticación TLS mutua	Sí	Sí

Características de seguridad	API de REST	API HTTP
Certificados para autenticación de backend	Sí	No
AWS WAF	Sí	No

Autorización

API Gateway admite varios mecanismos para controlar y administrar el acceso a la API. Para obtener más información, consulte the section called “Control de acceso” y the section called “Control de acceso”.

Opciones de autorización	API de REST	API HTTP
IAM	Sí	Sí
Políticas de recursos	Sí	No
Amazon Cognito	Sí	Sí [1]
Autorización personalizada con una función AWS Lambda	Sí	Sí
Token web JSON (JWT) [2]	No	Sí

[1]Puede utilizar Amazon Cognito con un autorizador de JWT.

[2]Puede utilizar un autorizador de Lambda para validar JWT para las API de REST.

Administración de API

Elija las API de REST si necesita capacidades de administración de API, como claves de API y limitación de velocidad por cliente. Para obtener más información, consulte the section called “Distribuir”, the section called “Nombres de dominio personalizados” y the section called “Nombres de dominio personalizados”.

Características	API de REST	API HTTP
Dominios personalizados	Sí	Sí
Claves de API	Sí	No
Limitación de velocidad por cliente	Sí	No
Limitación de uso por cliente	Sí	No
Portal para desarrolladores	Sí	No

Desarrollo

A medida que se desarrolla la API de API Gateway, se decide sobre una serie de características de la API. Estas características dependen del uso de la API. Para obtener más información, consulte the section called “Desarrollo” y the section called “Desarrollo”.

Características	API de REST	API HTTP
Configuración de CORS	Sí	Sí
Invocaciones de prueba	Sí	No
Almacenamiento en caché	Sí	No
Implementaciones controladas por el usuario	Sí	Sí
Implementaciones automáticas	No	Sí
Respuestas de gateway personalizadas	Sí	No
Implementación de la versión de valor controlado	Sí	No

Características	API de REST	API HTTP
Validación de solicitudes	Sí	No
Transformación de los parámetros de solicitud	Sí	Sí
Transformación del cuerpo de la solicitud	Sí	No

Supervisión

API Gateway admite varias opciones para registrar solicitudes de API y supervisar las API. Para obtener más información, consulte the section called “Supervisión” y the section called “Supervisión”.

Característica	API de REST	API HTTP
Métricas de Amazon CloudWatch	Sí	Sí
Registros de acceso a Amazon CloudWatch Logs	Sí	Sí
Registros de acceso a Amazon Data Firehose	Sí	No
Registros de ejecución	Sí	No
AWS X-Ray Rastreo de	Sí	No

Integraciones

Las integraciones conectan la API de API Gateway a los recursos de backend. Para obtener más información, consulte the section called “Integraciones” y the section called “Integraciones”.

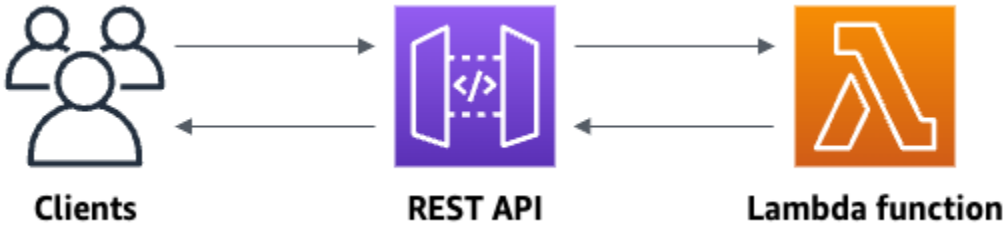
Característica	API de REST	API HTTP
Puntos de conexión HTTP públicos	Sí	Sí
AWS Servicios de	Sí	Sí
AWS Lambda functions	Sí	Sí
Integraciones privadas con equilibradores de carga de red	Sí	Sí
Integraciones privadas con equilibradores de carga de aplicaciones	Sí	Sí
Integraciones privadas con AWS Cloud Map	No	Sí
Integraciones simuladas	Sí	No
Transmisión de respuestas	Sí	No

Introducción a la consola de la API de REST

En este ejercicio de introducción, debe crear una API de REST sin servidor con la consola de API de REST de API Gateway. Las API sin servidor permiten centrarse en las aplicaciones, en lugar de dedicar el tiempo a aprovisionar y administrar servidores. Se debe completar este ejercicio en menos de 20 minutos y se debe realizar dentro del nivel gratuito de AWS.

En primer lugar, crea una función de Lambda con la consola de Lambda. A continuación, debe crear una API de REST mediante la consola de API de REST de la API Gateway. A continuación, se crea un método de API y se integra con una función de Lambda mediante una integración de proxy de Lambda. Por último, debe implementar e invocar la API.

Cuando invoca la API de REST, API Gateway enruta la solicitud a la función de Lambda. Lambda ejecuta la función y devuelve una respuesta a API Gateway. API Gateway, a continuación, le devuelve una respuesta.



Para completar este ejercicio, necesita una Cuenta de AWS y un usuario de AWS Identity and Access Management (IAM) con acceso a la consola. Para obtener más información, consulte Configuración para usar API Gateway.

Temas

- Paso 1: Crear una función Lambda
- Paso 2: Crear una API de REST
- Paso 3: Crear una integración de proxy de Lambda
- Paso 4: Implementar la API
- Paso 5: Invocar la API
- (Opcional) Paso 6: limpiar

Paso 1: Crear una función Lambda

Usa una función de Lambda para el backend de su API. Lambda ejecuta su código solo cuando es necesario y escala de manera automática, desde unas pocas solicitudes por día hasta miles por segundo.

Para este ejercicio, debe usar la función Node.js predeterminada en la consola de Lambda.

Cómo crear una función de Lambda

1. Inicie sesión en la consola de Lambda en https://console.aws.amazon.com/lambda/.
2. Elija Create function (Crear función).
3. En Basic information (Información básica), para Function name (Nombre de función), escriba **`my-function`**.
4. Para todas las demás opciones, utilice la configuración predeterminada.
5. Seleccione Creación de función.

El código predeterminado de la función de Lambda debe ser similar al siguiente:

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

Puede modificar la función de Lambda para este ejercicio, siempre y cuando la respuesta de la función se alinee con el formato que requiere API Gateway.

Sustituya el cuerpo de respuesta predeterminado (`Hello from Lambda!`) por `The API Gateway REST API console is great!`. Al invocar la función de ejemplo, devuelve una respuesta `200` a los clientes, junto con la respuesta actualizada.

Paso 2: Crear una API de REST

A continuación, debe crear una API de REST con un recurso raíz (/).

Para crear una API de REST

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Realice una de las siguientes acciones:
   - Para crear la primera API, para API de REST, elija Crear.
   - Si ha creado una API antes, elija Crear API y, a continuación, elija Crear para API de REST.
3. En API name (Nombre de la API), escriba**`my-rest-api`**.
4. (Opcional) En Description (Descripción), introduzca una descripción.
5. Mantenga Tipo de punto de conexión de la API establecido en Regional.
6. En Tipo de dirección IP, seleccione IPv4.
7. Seleccione Crear API.

Paso 3: Crear una integración de proxy de Lambda

A continuación, debe crear un método de API para la API de REST en el recurso raíz (/) e integrar el método con la función de Lambda mediante una integración de proxy. En una integración de proxy de Lambda, API Gateway pasa la solicitud entrante del cliente directamente a la función de Lambda.

Para crear una integración de proxy de Lambda

1. Seleccione el recurso `/` y, a continuación, elija Crear método.
2. En Tipo de método, seleccione `ANY`.
3. En Tipo de integración, seleccione Lambda.
4. Active Integración de proxy de Lambda.
5. En Función de Lambda, ingrese **`my-function`** y, continuación, seleccione la función de Lambda.
6. Elija Crear método.

Paso 4: Implementar la API

Después, debe crear una implementación de la API y asociarla con una etapa.

Para implementar su API

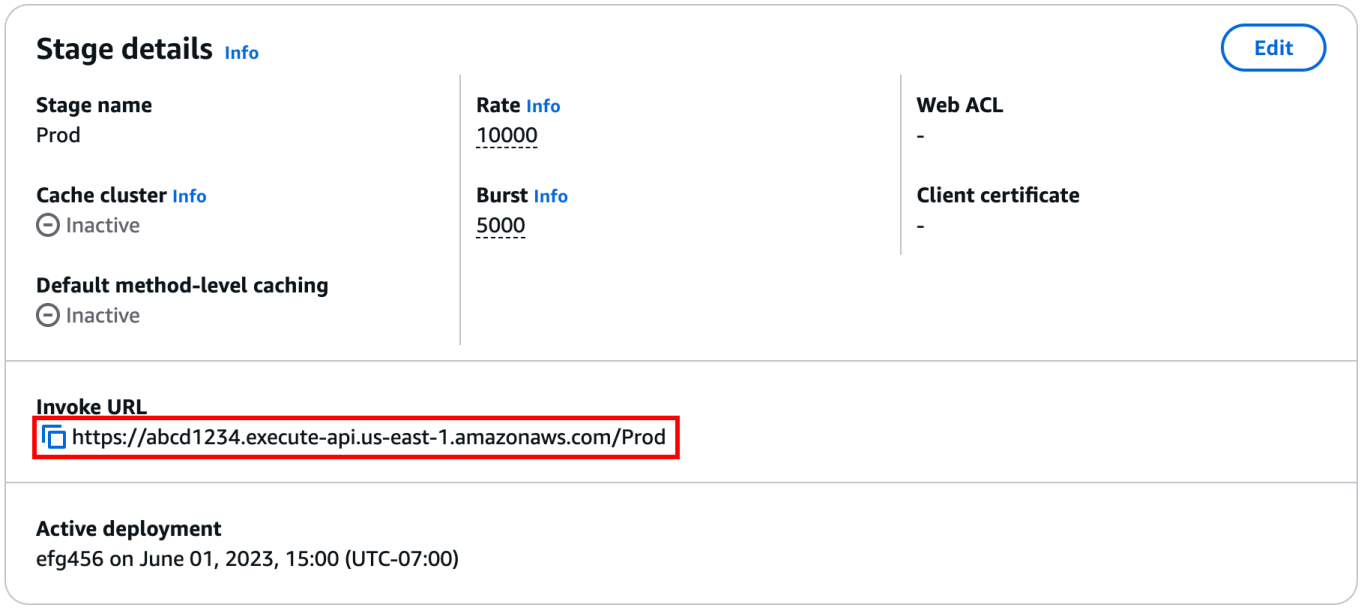
1. Elija Implementar API.
2. En Etapa, seleccione Nueva etapa.
3. En Stage name (Nombre de etapa), escriba **`Prod`**.
4. (Opcional) En Description (Descripción), introduzca una descripción.
5. Elija Implementar.

Ahora los clientes pueden llamar a la API. Para probar la API antes de implementarla, puede elegir, si lo desea, el método ANY, ir a la pestaña Probar y, a continuación, elegir Probar.

Paso 5: Invocar la API

Para invocar la API

1. En el panel de navegación principal, elija Etapa.
2. En Detalles de la etapa, elija el icono de copia para copiar la URL de invocación de la API.



3. Ingrese la URL de invocación en un navegador web.

   La URL completa debería ser `https://abcd123.execute-api.us-east-2.amazonaws.com/Prod`.

   Su navegador envía una `GET` solicitud a la API.
4. Verifique la respuesta de la API. Debería ver el texto `"The API Gateway REST API console is great!"` en el navegador.

(Opcional) Paso 6: limpiar

Para evitar acumular costos innecesarios en la Cuenta de AWS, elimine los recursos creados como parte de este ejercicio. Los siguientes pasos eliminan la API de REST, la función de Lambda y los recursos asociados.

Para eliminar la API de REST

1. En el panel Recursos, elija Acciones de la API y Eliminar API.
2. En el cuadro de diálogo Eliminar API, escriba confirmar y después elija Eliminar.

Para eliminar la función de Lambda

1. Inicie sesión en la consola de Lambda en https://console.aws.amazon.com/lambda/.
2. En la página Funciones, seleccione la función. Elija Acciones, Eliminar.

3. En el cuadro de diálogo Eliminar 1 función, escriba **`delete`** y, a continuación, elija Eliminar.

Para eliminar el grupo de registro de la función de Lambda

1. Abra la página Log grouls (Grupos de registro) de la consola Amazon CloudWatch.
2. En la página Grupos de registro, seleccione el grupo de registro de la función (`/aws/lambda/my-function`). A continuación, en Acciones, elija Eliminar grupos de registros.
3. En el cuadro de diálogo Eliminar grupos de registros, elija Eliminar.

Para eliminar el rol de ejecución de una función de Lambda

1. Abra la página Roles en la consola de IAM.
2. (Opcional) En la página Roles, en el cuadro de búsqueda, ingrese **`my-function`**.
3. Seleccione el rol de la función (por ejemplo, `my-function-`*`31exxmpl`*) y, a continuación, elija Eliminar.
4. En el cuadro de diálogo ¿Eliminar **`my-function-`*`31exxmpl`***?, escriba el nombre del rol y, a continuación, elija Eliminar.

> **Tip**
>
> Puede automatizar la creación y la limpieza de los recursos de AWS mediante el uso de AWS CloudFormation o AWS Serverless Application Model (AWS SAM). Para ver algunas plantillas de ejemplo de CloudFormation, consulte las plantillas de ejemplo de API Gateway en el repositorio GitHub de awsdocs.

Configuración para usar API Gateway

Antes de usar Amazon API Gateway por primera vez, complete las siguientes tareas.

Cómo crear una Cuenta de AWS

Para empezar a utilizar AWS, necesita una Cuenta de AWS. Para obtener más información sobre cómo crear una Cuenta de AWS, consulte la sección Introducción a una Cuenta de AWS en la Guía de referencia de AWS Account Management.

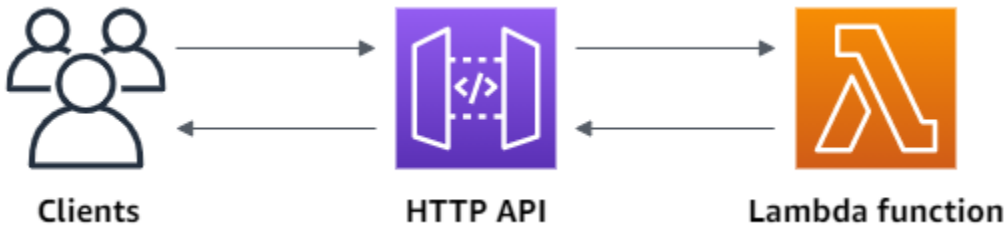
Introducción a API Gateway

En este ejercicio introductorio, creará una API sin servidor. Las API sin servidor permiten centrarse en las aplicaciones, en lugar de dedicar tiempo a aprovisionar y administrar servidores. Puede completar este ejercicio en menos de 20 minutos y realizarlo dentro del nivel gratuito de AWS.

En primer lugar, crea una función de Lambda con la consola de AWS Lambda. A continuación, crea una API HTTP mediante la consola de API Gateway. Luego, invoca su API.

> **Note**
>
> En este ejercicio se utiliza una API HTTP. API Gateway también es compatible con las API de REST, que incluyen más funciones. Para ver un tutorial sobre el uso de una API de REST, consulte the section called “Introducción a la consola de la API de REST”.
> Para obtener más información sobre las diferencias entre las API HTTP y las API de REST, consulte the section called “Elección entre las API de REST y las API de HTTP ”.

Cuando invoca su API HTTP, API Gateway enruta la solicitud a su función de Lambda. Lambda ejecuta la función de Lambda y devuelve una respuesta a API Gateway. API Gateway, a continuación, le devuelve una respuesta.



Para completar este ejercicio, necesita una cuenta de AWS y un usuario de AWS Identity and Access Management con acceso a la consola. Para obtener más información, consulte Requisitos previos.

Temas

- Paso 1: Crear una función Lambda
- Paso 2: crear una API HTTP
- Paso 3: probar la API
- (Opcional) Paso 4: limpiar
- Pasos siguientes

Paso 1: Crear una función Lambda

Usa una función de Lambda para el backend de su API. Lambda ejecuta su código solo cuando es necesario y escala de manera automática, desde unas pocas solicitudes por día hasta miles por segundo.

Para este ejemplo, utilice la función Node.js predeterminada de la consola de Lambda.

Para crear una función Lambda, realice el siguiente procedimiento:

1. Inicie sesión en la consola de Lambda en https://console.aws.amazon.com/lambda/.
2. Elija Create function (Crear función).
3. En Function name (Nombre de función), introduzca **`my-function`**.
4. Para todas las demás opciones, utilice la configuración predeterminada.
5. Elija Create function (Crear función).

La función de ejemplo devuelve una `200` respuesta a los clientes y el texto `Hello from Lambda!`.

Puede modificar su función de Lambda, siempre y cuando la respuesta de la función se alinea con el formato que requiere API Gateway.

El código predeterminado de la función de Lambda debe ser similar al siguiente:

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

Paso 2: crear una API HTTP

A continuación, cree una API HTTP. API Gateway también admite API REST y API WebSocket, pero una API HTTP es la mejor opción para este ejercicio. Las API de REST son compatibles con más funciones que las API HTTP, pero no las necesitamos para este ejercicio. Las API HTTP están diseñadas con características mínimas para que puedan ofrecerse a un precio más bajo. Las

API WebSocket mantienen conexiones persistentes con los clientes para la comunicación dúplex completo, lo cual no es necesario para este ejemplo.

La API HTTP proporciona un punto de enlace HTTP para su función de Lambda. API Gateway enruta las solicitudes a su función de Lambda y, a continuación, devuelve la respuesta de la función a los clientes.

Para crear una API HTTP

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Aplique alguna de las siguientes acciones:
   - Para crear su primera API, para HTTP API (API HTTP), elija Build (Crear).
   - Si ha creado una API antes, elija Create API (Crear API) y, a continuación, elija Build (Crear) para HTTP API (API HTTP).
3. Para Integrations (Integraciones), elija Add integration (Agregar integración).
4. Elija Lambda.
5. En Función Lambda, introduzca **`my-function`**.
6. En API name (Nombre de la API), escriba**`my-http-api`**.
7. Elija Next (Siguiente).
8. Revise la ruta que API Gateway crea para usted y, a continuación, elija Next (Siguiente).
9. Revise la etapa que API Gateway crea para usted y, a continuación, elija Next (Siguiente).
10. Seleccione Create (Crear).

Ahora ha creado una API HTTP con una integración de Lambda que está lista para recibir solicitudes de clientes.

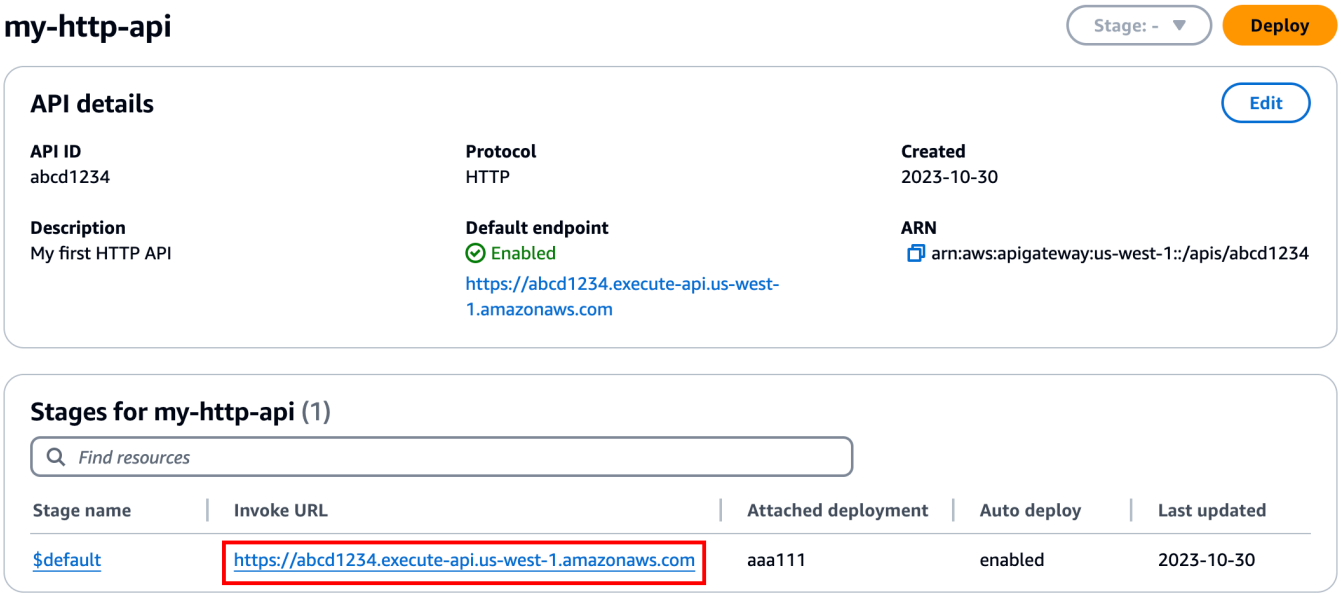
Paso 3: probar la API

A continuación, pruebe su API para asegurarse de que se encuentra en funcionamiento. Para mayor simplicidad, utilice un navegador web para invocar la API.

Para probar la API

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija la API.

3. Tenga en cuenta la URL de invocación de la API.



4. Copie la URL de invocación de la API y escríbala en un navegador web. Agregue el nombre de la función de Lambda a la URL de invocación para llamar a su función de Lambda. De forma predeterminada, la consola de API Gateway crea una ruta con el mismo nombre que su función de Lambda, `my-function`.

   La URL completa debería ser `https://`*`abcdef123`*`.execute-api.`*`us-east-2`*`.amazonaws.com/`*`my-function`*.

   Su navegador envía una `GET` solicitud a la API.

5. Verifique la respuesta de la API. Debería ver el texto `"Hello from Lambda!"` en el navegador.

(Opcional) Paso 4: limpiar

Para evitar costos innecesarios, elimine los recursos creados como parte de este ejercicio introductorio. Los siguientes pasos eliminan la API HTTP, la función de Lambda y los recursos asociados.

Para eliminar una API HTTP

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. En la página API, seleccione una API. Seleccione Actions y, luego, Delete.

3. Elija Eliminar.

Para eliminar una función de Lambda

1. Inicie sesión en la consola de Lambda en https://console.aws.amazon.com/lambda/.
2. En la página Functions (Funciones), seleccione una función. Seleccione Actions y, luego, Delete.
3. Elija Eliminar.

Para eliminar el grupo de registro de una función de Lambda

1. En la consola de Amazon CloudWatch, abra la página de grupos de registro.
2. En la página Grupos de registro, seleccione el grupo de registro de la función (`/aws/lambda/my-function`). Elija Actions (Acciones) y, a continuación, elija Delete log group (Eliminar grupo de registro).
3. Elija Eliminar.

Para eliminar el rol de ejecución de una función de Lambda

1. En la consola de AWS Identity and Access Management, abra la página Roles (Roles).
2. Seleccione el rol de la función, por ejempl, `my-function-31exxmpl`.
3. Elija Delete role (Eliminar rol).
4. Elija Sí, eliminar.

Puede automatizar la creación y la limpieza de los recursos de AWS mediante el uso de CloudFormation o AWS SAM. Para obtener plantillas de CloudFormation de ejemplo, consulte las plantillas de CloudFormation de ejemplo.

Pasos siguientes

Para este ejemplo, se utilizó la Consola de administración de AWS para crear una API HTTP simple. La API HTTP invoca una función de Lambda y devuelve una respuesta a los clientes.

A continuación, se indican los pasos siguientes a medida que continúa trabajando con API Gateway.

- Configure tipos adicionales de integraciones de API, entre los que se incluyen:

  - Puntos de enlace HTTP
  - Recursos privados en una VPC, como los servicios Amazon ECS
  - Servicios de AWS como Amazon Simple Queue Service, AWS Step Functions y Kinesis Data Streams
- Controlar el acceso a las API
- Habilitar el registro de las API
- Configurar la limitación de las API
- Configurar dominios personalizados para las API

Para obtener ayuda de la comunidad con Amazon API Gateway, consulte el foro de debate de API Gateway. Cuando ingrese a este foro, es posible que AWS requiera que inicie sesión.

Para obtener ayuda sobre API Gateway directamente de AWS, consulte las distintas opciones de soporte que se ofrecen en la página de AWS Support.

Consulte también nuestras preguntas frecuentes o contacte con nosotros directamente.

Tutoriales y talleres sobre Amazon API Gateway

Los siguientes tutoriales y talleres proporcionan ejercicios prácticos para ayudar a obtener información sobre API Gateway. Para comenzar, le recomendamos que complete the section called "Tutorial: Creación de una API de REST con una integración de proxy de Lambda".

Tutoriales sobre API REST

- Elección de un tutorial de integración de AWS Lambda
- Tutorial: Crear una API de REST importando un ejemplo
- Elección de un tutorial de integración de HTTP
- Tutorial: Creación de una API de REST con una integración privada
- Tutorial: Creación de una API de REST con una integración de AWS
- Tutorial: Creación de una API de REST de calculadora con dos integraciones de servicios de AWS y una integración de Lambda sin proxy
- Tutorial: Creación de una API de REST como proxy de Amazon S3
- Tutorial: Creación de una API de REST como proxy de Amazon Kinesis
- Tutorial: Creación de una API de REST mediante SDK de AWS o la AWS CLI
- Tutorial: Creación de una API de REST privada

Tutoriales sobre API HTTP

- Tutorial: Creación de una API de HTTP CRUD con Lambda y DynamoDB
- Tutorial: Creación de una API de HTTP con una integración privada en un servicio de Amazon ECS

Tutoriales de la API de WebSocket

- Tutorial: Creación de una aplicación de chat de WebSocket con una API de WebSocket, Lambda y DynamoDB

Talleres

- Desarrollar una aplicación web sin servidor

- CI/CD para aplicaciones sin servidor
- Taller sobre seguridad sin servidor
- Administración, autenticación y autorización de identidades sin servidor
- Taller sobre Amazon API Gateway

Tutoriales sobre API REST de Amazon API Gateway

Los siguientes tutoriales proporcionan ejercicios prácticos para ayudar a saber más sobre las API REST de API Gateway.

Temas

- Elección de un tutorial de integración de AWS Lambda
- Tutorial: Crear una API de REST importando un ejemplo
- Elección de un tutorial de integración de HTTP
- Tutorial: Creación de una API de REST con una integración privada
- Tutorial: Creación de una API de REST con una integración de AWS
- Tutorial: Creación de una API de REST de calculadora con dos integraciones de servicios de AWS y una integración de Lambda sin proxy
- Tutorial: Creación de una API de REST como proxy de Amazon S3
- Tutorial: Creación de una API de REST como proxy de Amazon Kinesis
- Tutorial: Creación de una API de REST mediante SDK de AWS o la AWS CLI
- Tutorial: Creación de una API de REST privada

Elección de un tutorial de integración de AWS Lambda

Para desarrollar una API con integraciones de Lambda puede utilizar la integración de proxy de Lambda o la integración de Lambda no de proxy.

En la integración de proxy de Lambda, la entrada para la función de Lambda se puede expresar como cualquier combinación de encabezados de solicitud, variables de ruta, parámetros de cadena de consulta, cuerpo y datos de configuración de API. Solo tiene que elegir una función de Lambda. API Gateway configura la solicitud de integración y la respuesta de integración por usted. Una vez configurado, el método de la API puede evolucionar sin modificar la configuración existente. Esto

es posible porque la función de Lambda del backend analiza los datos de la solicitud entrante y responde al cliente.

En la integración de Lambda no de proxy, debe asegurarse de que la entrada a la función de Lambda se proporcione como la carga de solicitud de integración. Debe asignar los datos de entrada que el cliente proporcionó como parámetros de solicitud en el cuerpo de solicitud de integración adecuado. Puede que también tenga que traducir el cuerpo de la solicitud proporcionado por el cliente a un formato reconocido por la función de Lambda.

En una integración de proxy de Lambda o sin proxy de Lambda, puede usar una función de Lambda en una cuenta diferente de la cuenta en la que creó la API.

Temas

- Tutorial: Creación de una API de REST con una integración de proxy de Lambda
- Tutorial: Creación de una API de REST con una integración no de proxy de Lambda
- Tutorial: Creación de una API de REST con integración de proxy de Lambda entre cuentas

Tutorial: Creación de una API de REST con una integración de proxy de Lambda

La integración de proxy de Lambda es un tipo de integración de API de API Gateway ligera y flexible que le permite integrar un método de API, o la totalidad de la API, con una función de Lambda. La función de Lambda se puede escribir en cualquier lenguaje que admitida Lambda. Debido a que se trata de una integración de proxy, puede cambiar la implementación de la función de Lambda en cualquier momento sin tener que volver a implementar la API.

En este tutorial, aprenderá a hacer lo siguiente:

- Creación de una función Hello, World! Función de Lambda para que sea el backend de la API.
- Creación y prueba de una función "Hello, World!" API con integración de proxy de Lambda.

Temas

- Creación de una función Hello, World! Función de Lambda
- Creación de una función Hello, World! API
- Implementación y pruebas de API

Creación de una función Hello, World! Función de Lambda

Para crear un "Hello, World!" Función de Lambda en la consola de Lambda

1. Inicie sesión en la consola de Lambda en https://console.aws.amazon.com/lambda/.
2. En la barra de navegación de AWS, elija una Región de AWS.

   > **Note**
   >
   > Anote la región en la que ha creado la función de Lambda. La necesitará al crear la API.

3. Elija Functions (Funciones) en el panel de navegación.
4. Elija Create function (Crear función).
5. Elija Crear desde cero.
6. Bajo Basic information (Información básica), haga lo siguiente:

   a. Bajo Function name (Nombre de función), escriba **`GetStartedLambdaProxyIntegration`**.

   b. En Tiempo de ejecución, elija el último tiempo de ejecución de Node.js o Python compatible.

   c. Para Arquitectura, mantenga la configuración predeterminada.

   d. En Permissions (Permisos), expanda Change default execution role(Cambiar rol de ejecución predeterminado). En Rol de ejecución, elija Crear un nuevo rol desde las plantillas de políticas de AWS.

   e. En Role name (Nombre del rol), escriba **`GetStartedLambdaBasicExecutionRole`**.

   f. Deje el campo Policy templates (Plantillas de política) en blanco.

   g. Elija Create function (Crear función).

7. En Function code (Código de la función), en el editor de código integrado, copie/pegue el código siguiente:

   Node.js

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

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

Python

```
import json


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

    greeter = 'World'

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

```
    try:
        if (event['multiValueHeaders']) and (event['multiValueHeaders']
['greeter']) and (
                event['multiValueHeaders']['greeter'] is not None):
            greeter = " and ".join(event['multiValueHeaders']['greeter'])
    except KeyError:
        print('No greeter')

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

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

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

    return res
```

8. Elija Deploy (Implementar).

Creación de una función Hello, World! API

Ahora cree una API para la función de Lamba "Hello, World!" La función de Lambda utilizando la consola de API Gateway.

Para crear un "Hello, World!" API

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.

2. Si es la primera vez que utiliza API Gateway, verá una página en la que aparecen las características del servicio. En REST API, elija Build (Compilación). Cuando aparezca el menú emergente Create Example API (Crear API de ejemplo), elija OK (Aceptar).

   Si esta no es la primera vez que utiliza API Gateway, elija Create API (Crear API). En REST API, elija Build (Compilación).
3. En API name (Nombre de la API), escriba**`LambdaProxyAPI`**.
4. (Opcional) En Description (Descripción), introduzca una descripción.
5. Mantenga Tipo de punto de conexión de la API establecido en Regional.
6. En Tipo de dirección IP, seleccione IPv4.
7. Seleccione Crear API.

Después de crear una API, se crea un recurso. Normalmente, los recursos de la API están organizados en un árbol de recursos de acuerdo con la lógica de la aplicación. Para este ejemplo, creará un recurso /helloworld.

Para crear un recurso

1. Elija Crear recurso.
2. Mantenga Recurso proxy desactivado.
3. Mantenga Ruta del recurso en `/`.
4. En Nombre del recurso, escriba **`helloworld`**.
5. Mantenga desactivado CORS (uso compartido de recursos entre orígenes).
6. Elija Crear recurso.

En una integración de proxy, toda la solicitud se envía a la función de Lambda del backend tal y como está, a través de un método catch-all `ANY` que representa cualquier método HTTP. El método HTTP real lo especifica el cliente en tiempo de ejecución. El método `ANY` le permite usar una sola configuración de métodos de API para todos los métodos HTTP admitidos: `DELETE`, `GET`, `HEAD`, `OPTIONS`, `PATCH`, `POST` y `PUT`.

Para crear un método **`ANY`**

1. Seleccione el recurso /helloworld y, a continuación, elija Crear método.
2. En Tipo de método, seleccione CUALQUIERA.

3. En Tipo de integración, seleccione Función de Lambda.
4. Active Integración de proxy de Lambda.
5. En Función de Lambda, seleccione la Región de AWS en la que creó la función de Lambda y, a continuación, introduzca el nombre de la función.
6. Si desea utilizar el valor predeterminado del tiempo de espera, que es de 29 segundos, mantenga activado el Tiempo de espera predeterminado. Para establecer un tiempo de espera personalizado, elija Tiempo de espera predeterminado e ingrese un valor de tiempo de espera comprendido entre `50` y `29000` milisegundos.
7. Elija Crear método.

Implementación y pruebas de API

Para implementar su API

1. Elija Implementar API.
2. En Etapa, seleccione Nueva etapa.
3. En Stage name (Nombre de etapa), escriba **`test`**.
4. (Opcional) En Description (Descripción), introduzca una descripción.
5. Elija Implementar.
6. En Detalles de la etapa, elija el icono de copia para copiar la URL de invocación de la API.

Uso del navegador y cURL para probar una API con integración de proxy de Lambda

Puede utilizar un navegador o cURL para probar la API.

Para probar solicitudes `GET` mediante solo los parámetros de cadena de consulta, puede ingresar la URL para el recurso `helloworld` de la API en la barra de direcciones del navegador.

Para crear la URL del recurso `helloworld` de la API, agregue el recurso `helloworld` y el parámetro de cadena de consulta `?greeter=John` a la URL de invocación. La URL debería tener el siguiente aspecto.

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

Para otros métodos, debe utilizar utilidades de prueba de la API de REST más avanzadas, tales como POSTMAN o cURL. En este tutorial se utiliza cURL. Los ejemplos de comandos de cURL que aparecen a continuación se presupone que cURL está instalado en su equipo.

Para probar la API implementada mediante cURL:

1. Abra una ventana de terminal.
2. Copie el siguiente comando cURL y péguelo en la ventana de terminal. Sustituya la URL de invocación por la que copió en el paso anterior y añada **`/helloworld`** al final de la URL.

> **Note**
>
> Si está ejecutando el comando en Windows, utilice esta sintaxis:
>
> ```
> curl -v -X POST "https://r275xc9bmd.execute-api.us-east-1.amazonaws.com/
> test/helloworld" -H "content-type: application/json" -d "{ \"greeter\":
>  \"John\" }"
> ```

   a. Para llamar a la API con el parámetro de cadena de consulta de `?greeter=John`:

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

   b. Para llamar a la API con el parámetro de encabezado de `greeter:John`:

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

   c. Para llamar a la API con un cuerpo de `{"greeter":"John"}`:

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

   En todos estos casos, la salida es una respuesta 200 con el siguiente cuerpo de respuesta:

```
Hello, John!
```

Tutorial: Creación de una API de REST con una integración no de proxy de Lambda

En este tutorial utilizaremos la consola de API Gateway para crear una API que permita a un cliente llamar a funciones de Lambda a través de la integración de Lambda no de proxy (conocida también como integración personalizada). Para obtener más información acerca de las funciones de AWS Lambda y de Lambda, consulte la Guía para desarrolladores de AWS Lambda.

Para facilitar el aprendizaje, elegimos una función de Lambda sencilla con una configuración de API mínima para guiarlo para desarrollar una API de API Gateway con la integración de Lambda personalizada. Cuando sea necesario, describiremos parte de la lógica. Para ver un ejemplo más detallado de la integración de Lambda personalizada, consulte Tutorial: Creación de una API de REST de calculadora con dos integraciones de servicios de AWS y una integración de Lambda sin proxy.

Antes de crear la API, configure el backend de Lambda mediante la creación de una función de Lambda en AWS Lambda, lo cual se describe a continuación.

Temas

- Creación de una función de Lambda para la integración de Lambda no de proxy
- Creación de una API con la integración de Lambda no de proxy
- Prueba que invoca el método de API
- Implementar la API
- Pruebe la API en una etapa de implementación
- Eliminar recursos

Creación de una función de Lambda para la integración de Lambda no de proxy

> **Note**
>
> La creación de funciones de Lambda puede suponer cargos en su cuenta de AWS.

En este paso creará una función de Lambda del tipo "Hello, World!" para la integración personalizada de Lambda. En esta tutorial, la función se denomina `GetStartedLambdaIntegration`.

La implementación de esta función de Lambda `GetStartedLambdaIntegration` es la siguiente:

Node.js

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

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

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

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

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

Python

```
import json

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

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


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

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

Para la integración de Lambda personalizada, API Gateway transmite la entrada a la función de Lambda del cliente como cuerpo de solicitud de integración. El objeto `event` de gestión de la función de Lambda es la entrada.

Nuestra función de Lambda es simple. Analiza el objeto de entrada `event` para las propiedades `name`, `city`, `time` y `day`. Entonces, devuelve un saludo, como un objeto JSON de `{"message":greeting}`, al intermediario. El mensaje se encuentra en el patrón `"Good [morning|afternoon|day], [name|you] in [city|World]. Happy day!"`. Se supone que la entrada a la función de Lambda es del siguiente objeto JSON:

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

Para obtener más información, consulte AWS Lambda Developer Guide.

Además, la función registra su ejecución en Amazon CloudWatch llamando a `console.log(...)`. Esto es útil para rastrear llamadas cuando se depura la función. Para permitir que la función `GetStartedLambdaIntegration` registre la llamada, establezca un rol de IAM con las políticas apropiadas para la función de Lambda para crear los flujos de CloudWatch y agregar entradas de registro a los flujos. La consola de Lambda lo guía para crear los roles y políticas de IAM requeridos.

Si configura la API sin utilizar la consola de API Gateway (por ejemplo, si importa una API de un archivo de OpenAPI), tendrá que crear de forma explícita, si es necesario, y configurar un rol y una política de invocación para que API Gateway invoque las funciones de Lambda. Para obtener más información sobre cómo configurar los roles de ejecución e invocación de Lambda para una API de API Gateway, consulte Control del acceso a una API de REST con permisos de IAM.

En comparación con `GetStartedLambdaProxyIntegration`, la función de Lambda para la integración de proxy de Lambda, la función de Lambda `GetStartedLambdaIntegration` para la integración personalizada de Lambda solo toma entradas del cuerpo de solicitud de integración de API de API Gateway. La función puede devolver una salida de cualquier objeto JSON, una cadena, un número, un booleano o incluso un blob binario. Por el contrario, la función de Lambda de la integración de proxy de Lambda puede tomar la entrada de cualquier dato solicitado, pero debe devolver una salida de un objeto JSON específico. La función `GetStartedLambdaIntegration` de la integración de Lambda personalizada puede tener los parámetros de solicitud de API como entrada, siempre que API Gateway asigne los parámetros de solicitud de API requeridos al cuerpo de solicitud de la integración antes de reenviar la solicitud del cliente al backend. Para que esto ocurra, el desarrollador de la API debe crear una plantilla de mapeo y debe configurarla en el método de API durante la creación de la API.

Ahora, cree la función de Lambda `GetStartedLambdaIntegration`.

Para crear la función de Lambda **`GetStartedLambdaIntegration`** para la integración personalizada de Lambda

1. Abra la consola de AWS Lambda en https://console.aws.amazon.com/lambda/.
2. Realice una de las siguientes acciones:
   - Si aparece la página de bienvenida, elija Get Started Now (Empezar ahora) y, a continuación, elija Create function (Crear función).
   - Si aparece la página de lista Lambda > Functions (Lambda > Funciones), elija Create function (Crear función).
3. Elija Author from scratch.
4. En el panel Author from scratch (Crear desde cero), haga lo siguiente:
   a. En Name (Nombre), escriba **`GetStartedLambdaIntegration`** como nombre de la función de Lambda.
   b. En Tiempo de ejecución, elija el último tiempo de ejecución de Node.js o Python compatible.
   c. Para Arquitectura, mantenga la configuración predeterminada.
   d. En Permissions (Permisos), expanda Change default execution role(Cambiar rol de ejecución predeterminado). En Rol de ejecución, elija Crear un nuevo rol desde las plantillas de políticas de AWS.
   e. En Role name (Nombre de la función), escriba un nombre para la función (por ejemplo, **`GetStartedLambdaIntegrationRole`**).
   f. En Policy templates (Plantillas de política), elija Simple microservice permissions (Permisos para microservicios sencillos).
   g. Elija Create function (Crear función).
5. En el panel Configure function (Configurar función), en Function code (Código de función), configure los campos siguientes:
   a. Copie el código de la función de Lambda que aparece al comienzo de esta sección y péguelo en el editor del código en línea.
   b. Deje las opciones predeterminadas para todos los demás campos en esta sección.
   c. Elija Deploy (Implementar).
6. Para probar la función recién creada, seleccione la pestaña Probar.
   a. En Nombre del evento, escriba **`HelloWorldTest`**.

b. En JSON de evento, sustituya el código predeterminado por lo siguiente.

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

c. Elija Test (Probar) para invocar la función. Se muestra la sección Execution result: succeeded (Resultado de ejecución: realizada correctamente). Expanda Detalles y verá el siguiente resultado.

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

La salida también se escribe en CloudWatch Logs.

Como ejercicio secundario, puede utilizar la consola de IAM para ver el rol de IAM (`GetStartedLambdaIntegrationRole`) que fue creado como parte de la creación de la función de Lambda. Este rol de IAM posee dos políticas en línea asociadas. Una de ellas estipula los permisos más básicos para la ejecución de Lambda. Permite llamar a `CreateLogGroup` de CloudWatch para cualquier recurso de CloudWatch de su cuenta en la región donde se crea la función de Lambda. Esta política también permite crear flujos de CloudWatch y registrar eventos para la función de Lambda `GetStartedLambdaIntegration`.

JSON

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

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

El documento de la otra política se aplica para invocar otro servicio de AWS que no se utiliza en este ejemplo. Puede omitirla por ahora.

Asociada con el rol de IAM hay una entidad de confianza, que es `lambda.amazonaws.com`. Aquí se encuentra la relación de confianza:

JSON

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

La combinación de esta relación de confianza y la política en línea hace que sea posible para la función de Lambda invocar una función `console.log()` para registrar los eventos en CloudWatch Logs.

Creación de una API con la integración de Lambda no de proxy

Con la función de Lambda (`GetStartedLambdaIntegration`) creada y probada, está listo para exponer la función a través de una API de API Gateway. Para fines ilustrativos, expondremos la función de Lambda con un método HTTP genérico. Utilizamos el cuerpo de solicitud, una variable de ruta de la URL, una cadena de consulta y un encabezado para recibir los datos de entrada requeridos del cliente. Activamos el validador de la solicitud de API Gateway para que la API asegure que todos los datos requeridos se definen y especifican de forma adecuada. Configuramos una plantilla de mapeo para que API Gateway transforme los datos de solicitud proporcionados por el cliente a un formato válido, según lo requiera la función de Lambda del backend.

Para crear una API con la integración de Lambda no de proxy

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Si es la primera vez que utiliza API Gateway, verá una página en la que aparecen las características del servicio. En REST API, elija Build (Compilación). Cuando aparezca el menú emergente Create Example API (Crear API de ejemplo), elija OK (Aceptar).

   Si esta no es la primera vez que utiliza API Gateway, elija Create API (Crear API). En REST API, elija Build (Compilación).
3. En API name (Nombre de la API), escriba**`LambdaNonProxyAPI`**.
4. (Opcional) En Description (Descripción), introduzca una descripción.
5. Mantenga Tipo de punto de conexión de la API establecido en Regional.
6. En Tipo de dirección IP, seleccione IPv4.
7. Seleccione Crear API.

Tras crear su API, cree un recurso /{city}. Este es un ejemplo de un recurso con una variable de ruta que toma una entrada del cliente. Más adelante, mapeará esta variable de ruta en la entrada de la función de Lambda mediante una plantilla de mapeo.

Para crear un recurso

1. Elija Crear recurso.
2. Mantenga Recurso proxy desactivado.
3. Mantenga Ruta del recurso en `/`.
4. En Nombre del recurso, escriba **`{city}`**.

5. Mantenga desactivado CORS (uso compartido de recursos entre orígenes).
6. Elija Crear recurso.

Tras crear su recurso /{city}, cree un método `ANY`. El verbo `ANY` de HTTP es un marcador para un método HTTP válido que un cliente envía en el tiempo de ejecución. Este ejemplo muestra que el método `ANY` se puede utilizar para la integración de Lambda personalizada y para la integración de proxy de Lambda.

Para crear un método **`ANY`**

1. Seleccione el recurso /{city} y, a continuación, elija Crear método.
2. En Tipo de método, seleccione CUALQUIERA.
3. En Tipo de integración, seleccione Función de Lambda.
4. Mantenga desactivada la Integración de proxy Lambda.
5. En Función de Lambda, seleccione la Región de AWS en la que creó la función de Lambda y, a continuación, introduzca el nombre de la función.
6. Elija Configuración de solicitud de método.

   Ahora, active un validador de la solicitud para una variable de ruta de la URL, un parámetro de cadena de consulta y un encabezado para garantizar que se hayan definido todos los datos necesarios. Para este ejemplo, se crea un parámetro de cadena de consulta `time` y un encabezado `day`.
7. En Validador de solicitud, seleccione Validar parámetros de cadena de consulta y encabezados.
8. Elija Parámetros de cadenas de consulta de URL y haga lo siguiente:

   a. Elija Añadir cadena de consulta.
   b. En Nombre, escriba **`time`**.
   c. Active la opción Obligatorio.
   d. Mantenga Almacenamiento en caché desactivado.
9. Elija Encabezados de solicitudes HTTP y haga lo siguiente:

   a. Elija Agregar encabezado.
   b. En Nombre, escriba **`day`**.
   c. Active la opción Obligatorio.
   d. Mantenga Almacenamiento en caché desactivado.

10. Elija Crear método.

Tras activar un validador de solicitudes, se configura la solicitud de integración para el método ANYañadiendo una plantilla body-mapping para transformar la solicitud entrante en una carga JSON, tal y como exige la función de Lambda de backend.

Para configurar la solicitud de integración

1. En la pestaña Solicitud de integración, en Configuración de solicitud de integración, elija Editar.
2. En Acceso directo de cuerpo de la solicitud, elija Cuando no haya plantillas definidas (recomendado).
3. Elija Plantillas de mapeo.
4. Elija Add mapping template (Añadir plantilla de asignación).
5. En Tipo de contenido, ingrese **`application/json`**.
6. En Cuerpo de la plantilla, introduzca el siguiente código:

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

7. Seleccione Save.

Prueba que invoca el método de API

La consola de API Gateway proporciona un servicio de pruebas para que pueda probar la invocación a la API antes de que se implemente. Puede utilizar la característica Prueba de la consola para probar la API mediante el envío de la siguiente solicitud:

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

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

En esta solicitud de prueba, configurará ANY en POST, establecerá {city} en Seattle, asignará Wednesday como el valor del encabezado day y asignará "John" como el valor callerName.

Para probar el método **ANY**

1. Elija la pestaña Prueba. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña.
2. En Tipo de método, seleccione POST.
3. En Ruta, debajo de ciudad, introduzca **Seattle**.
4. En Cadenas de consulta, escriba **time=morning**.
5. En Encabezados, escriba **day:Wednesday**.
6. En Cuerpo de la solicitud, introduzca **{ "callerName": "John" }**.
7. Seleccione Test (Probar).

Compruebe que la carga de respuesta devuelta sea la siguiente:

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

También puede ver los registros para ver cómo API Gateway procesa la solicitud y la respuesta.

```
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
```

Los registros muestran la solicitud entrante antes del mapeo y la solicitud de integración después del mapeo. Cuando una prueba falla, los registros son útiles para evaluar si la entrada original es correcta o si la plantilla de mapeo funciona correctamente.

Implementar la API

La invocación de prueba es una simulación que tiene limitaciones. Por ejemplo, elude cualquier mecanismo de autorización promulgado en la API. Para probar la ejecución de la API en tiempo real, primero debe implementar la API. Para implementar una API, usted crea una etapa para crear una snapshot de la API en ese momento. El nombre de etapa también define la ruta de base después del nombre de host predeterminado de la API. El recurso raíz de la API se agrega después del nombre

de la etapa. Al modificar la API, debe volver a implementarla a una etapa nueva o existente antes de que los cambios surtan efecto.

Para implementar la API en una etapa

1. Elija Implementar API.
2. En Etapa, seleccione Nueva etapa.
3. En Stage name (Nombre de etapa), escriba **`test`**.

   > **Note**
   >
   > La entrada debe tener texto cifrado UTF-8 (es decir, no localizado).

4. (Opcional) En Description (Descripción), introduzca una descripción.
5. Elija Implementar.

En Detalles de la etapa, elija el icono de copia para copiar la URL de invocación de la API. El patrón general de esta URL base de la API es `https://`*`api-id`*`.`*`region`*`.amazonaws.com/`*`stageName`*. Por ejemplo, la URL base de la API (`beags1mnid`) creada en la región `us-west-2` e implementada en la etapa `test` es `https://beags1mnid.execute-api.us-west-2.amazonaws.com/test`.

Pruebe la API en una etapa de implementación

Puede probar una API implementada de diferentes maneras. Para las solicitudes GET que solo utilizan variables de ruta de URL o parámetros de cadena de consulta, puede escribir la URL del recurso de la API en un explorador. Para otros métodos, debe utilizar utilidades de prueba de la API de REST más avanzadas, tales como POSTMAN o cURL.

Para probar la API mediante cURL

1. Abra una ventana de la terminal en su equipo local conectado a Internet.
2. Para probar `POST /Seattle?time=evening`:

   Copie el siguiente comando cURL y péguelo en la ventana de la terminal.

```
curl -v -X POST \
  'https://beags1mnid.execute-api.us-west-2.amazonaws.com/test/Seattle?
time=evening' \
  -H 'content-type: application/json' \
```

```
  -H 'day: Thursday' \
  -H 'x-amz-docs-region: us-west-2' \
  -d '{
 "callerName": "John"
}'
```

Debería recibir una respuesta correcta con la siguiente carga:

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

Si cambia `POST` a `PUT` en esta solicitud de método, recibe la misma respuesta.

Eliminar recursos

Si ya no necesita las funciones de Lambda que ha creado para este tutorial, puede eliminarlas ahora. También puede eliminar los recursos de IAM asociados.

> **Warning**
>
> Si tiene previsto completar el resto de tutoriales de esta serie, no elimine el rol de ejecución de Lambda ni el rol de invocación de Lambda. Si elimina una función de Lambda que usan sus API, esas API dejarán de funcionar. La eliminación de una función de Lambda no se puede deshacer. Si desea utilizar la función de Lambda de nuevo, debe volver a crearla.
> Si elimina un recurso IAM que usa una función de Lambda, esa función de Lambda dejará de funcionar y las API que dependen de esa función tampoco funcionarán. La eliminación de un recurso de IAM no se puede deshacer. Si desea utilizar el recurso de IAM de nuevo, debe volver a crearlo.

Para eliminar la función de Lambda

1. Inicie sesión en la Consola de administración de AWS y abra la consola de AWS Lambda en https://console.aws.amazon.com/lambda/.
2. En la lista de funciones, elija GetStartedLambdaIntegration, Acciones y, a continuación, Eliminar función. Cuando se le pregunte, elija Delete (Eliminar) otra vez.

Para eliminar los recursos de IAM asociados

1. Abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En Details (Detalles), elija Roles (Roles).
3. De entre la lista de roles, seleccione GetStartedLambdaIntegrationRole, elija Acciones del rol y, a continuación, Eliminar rol. Siga los pasos en la consola para eliminar el rol.

Tutorial: Creación de una API de REST con integración de proxy de Lambda entre cuentas

Ahora puede utilizar una función AWS Lambda desde otra cuenta de AWS como backend de integración de API. Cada cuenta puede estar en cualquier región en la que Amazon API Gateway esté disponible. Esto permite administrar y compartir de forma centralizada funciones de backend de Lambda en varias API.

En esta sección mostramos cómo configurar una integración de proxy de Lambda entre cuentas con la consola de Amazon API Gateway.

Creación de una API de API Gateway para la integración de Lambda entre cuentas

Para crear un API

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Si es la primera vez que utiliza API Gateway, verá una página en la que aparecen las características del servicio. En REST API, elija Build (Compilación). Cuando aparezca el menú emergente Create Example API (Crear API de ejemplo), elija OK (Aceptar).

   Si esta no es la primera vez que utiliza API Gateway, elija Create API (Crear API). En REST API, elija Build (Compilación).
3. En API name (Nombre de la API), escriba**`CrossAccountLambdaAPI`**.
4. (Opcional) En Description (Descripción), introduzca una descripción.
5. Mantenga Tipo de punto de conexión de la API establecido en Regional.
6. En Tipo de dirección IP, seleccione IPv4.
7. Seleccione Crear API.

Creación de una función de integración de Lambda en otra cuenta

Ahora, creará una función de Lambda en una cuenta distinta de la que usó para crear la API de ejemplo.

Creación de una función de Lambda en otra cuenta

1. Inicie sesión en la consola de Lambda de una cuenta distinta de la que utilizó para crear la API de API Gateway.
2. Elija Create function (Crear función).
3. Elija Author from scratch.
4. En Author from scratch (Crear desde cero), haga lo siguiente:
   a. En Function name (Nombre de función), escriba un nombre.
   b. En la lista desplegable Runtime (Tiempo de ejecución), elija un tiempo de ejecución de Node.js compatible.
   c. Para Arquitectura, mantenga la configuración predeterminada.
   d. En Permissions (Permisos), expanda Choose or create an execution role (Seleccionar o crear un rol de ejecución). Puede crear un rol o elegir uno existente.
   e. Elija Create function (Crear función) para continuar.
5. Desplácese hacia abajo en el panel Function code (Código de la función).
6. Introduzca la implementación de la función de Node.js desde the section called “Tutorial: Creación de una API de REST con una integración de proxy de Lambda”.
7. Elija Deploy (Implementar).
8. Anote el ARN completo de la función (se encuentra en la esquina superior derecha del panel de la función de Lambda). Lo necesitará al crear la integración de Lambda entre cuentas.

Configuración de la integración de Lambda entre cuentas

Una vez que tenga una función de integración de Lambda en otra cuenta, puede utilizar la consola de API Gateway para agregarla a la API de la primera cuenta.

> **Note**
>
> Si está configurando un autorizador entre regiones y cuentas, el valor `sourceArn` que se agrega a la función de destino debe utilizar la región de la función, no la región de la API.

Después de crear una API, se crea un recurso. Normalmente, los recursos de la API están organizados en un árbol de recursos de acuerdo con la lógica de la aplicación. Para este ejemplo, creará un recurso /helloworld.

Para crear un recurso

1. Elija Crear recurso.
2. Mantenga Recurso proxy desactivado.
3. Mantenga Ruta del recurso en `/`.
4. En Nombre del recurso, escriba **`helloworld`**.
5. Mantenga desactivado CORS (uso compartido de recursos entre orígenes).
6. Elija Crear recurso.

Después de crear un recurso, se crea un método `GET`. El método `GET` se integra con una función de Lambda en otra cuenta.

Para crear un método **`GET`**

1. Seleccione el recurso /helloworld y, a continuación, elija Crear método.
2. En Tipo de método, seleccione GET.
3. En Tipo de integración, seleccione Función de Lambda.
4. Active Integración de proxy de Lambda.
5. En Función de Lambda, introduzca el ARN completo de la función de Lambda del paso 1.

   En la consola de Lambda, puede encontrar el ARN de la función en la esquina superior derecha de la ventana de la consola.
6. Al introducir el ARN, aparecerá una cadena de comandos `aws lambda add-permission`. Esto concederá a su primera cuenta acceso a la función de Lambda de la segunda cuenta. Copie y pegue la cadena de comandos `aws lambda add-permission` en una ventana de la AWS CLI configurada para la segunda cuenta.
7. Elija Crear método.

Para ver la política actualizada para la función en la consola de Lambda.

(Opcional) Para ver la política actualizada

1. Inicie sesión en la Consola de administración de AWS y abra la consola AWS Lambda en https://console.aws.amazon.com/lambda/.
2. Elija su función de Lambda.
3. Elija Permissions.

   Debería ver una política `Allow` con una cláusula `Condition` en la que `AWS:SourceArn` es el ARN del método `GET` de la API.

Tutorial: Crear una API de REST importando un ejemplo

Puede utilizar la consola de Amazon API Gateway para crear y probar una API REST sencilla con integración HTTP para un sitio web PetStore. La definición de la API está preconfigurada como un archivo de OpenAPI 2.0. Después de cargar la definición de API en API Gateway, puede utilizar la consola de API Gateway para examinar la estructura básica de API o simplemente implementar y probar la API.

La API de ejemplo de PetStore admite los siguientes métodos para que un cliente obtenga acceso al sitio web de backend HTTP de `http://petstore-demo-endpoint.execute-api.com/petstore/pets`.

> **Note**
>
> En este tutorial se utiliza un punto de conexión HTTP como ejemplo. Cuando cree sus propias API, le recomendamos que utilice puntos de conexión HTTPS para las integraciones HTTP.

- `GET /`: para leer el acceso del recurso raíz de la API que no se integra con ningún punto de enlace de backend. API Gateway responde con información general del sitio web PetStore. Este es un ejemplo del tipo de integración `MOCK`.
- `GET /pets`: para obtener acceso de lectura al recurso `/pets` de la API que se integra con el recurso `/pets` de backend asignado. El backend devuelve una página de mascotas disponibles en PetStore. Este es un ejemplo del tipo de integración `HTTP`. La URL del punto de enlace de integración es `http://petstore-demo-endpoint.execute-api.com/petstore/pets`.

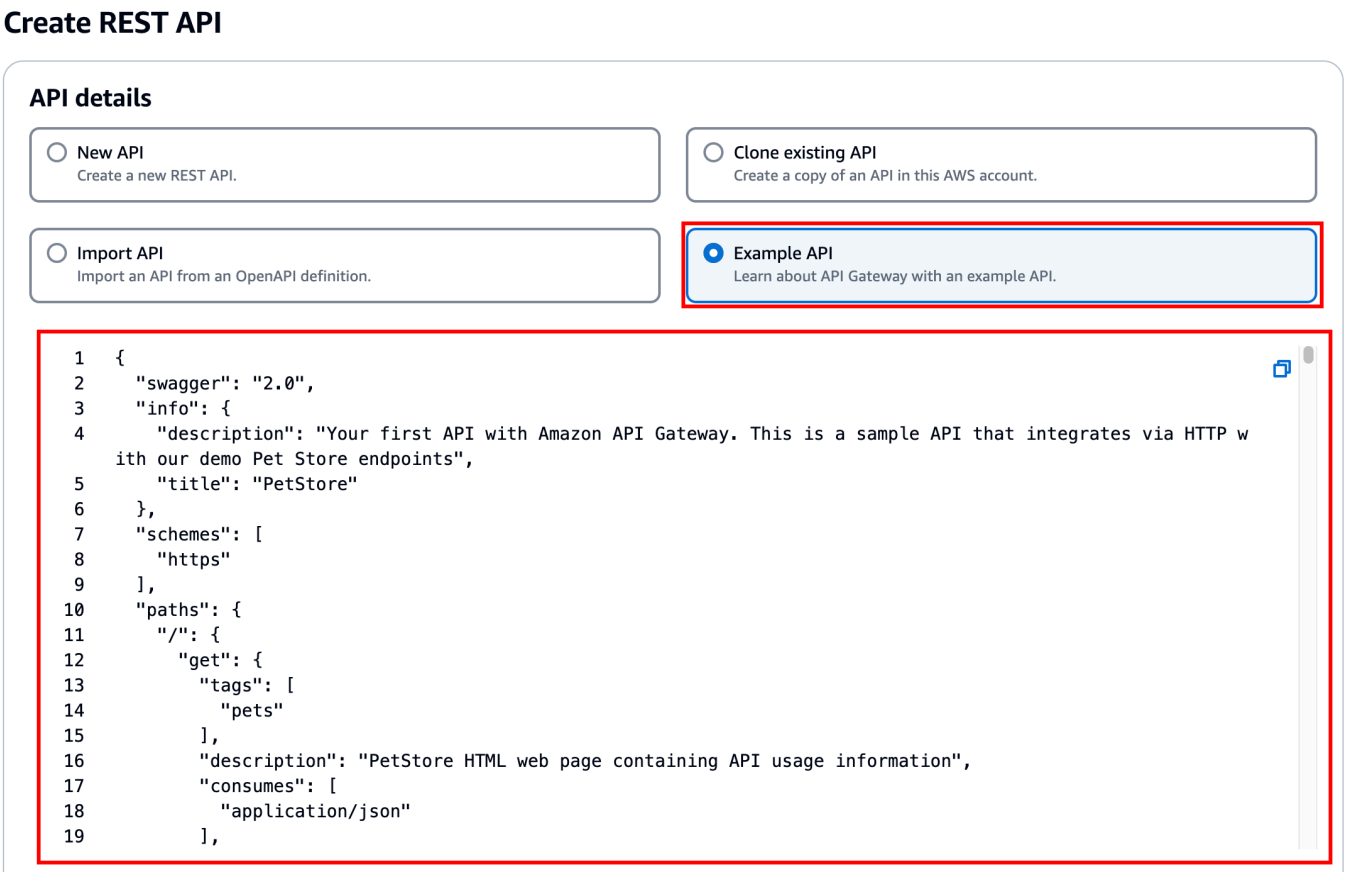
- `POST /pets`: para obtener acceso de escritura al recurso `/pets` de la API que se integra con el recurso `/petstore/pets` de backend. Tras recibir una solicitud correcta, el backend agrega la mascota especificada a PetStore y devuelve el resultado al autor de la llamada. La integración también es `HTTP`.
- `GET /pets/{petId}`: para obtener acceso de lectura a una mascota identificada por un valor `petId` tal como se especifica en una ruta variable de la URL de solicitud de entrada. Este método también tiene el tipo de integración `HTTP`. El backend devuelve la mascota especificada que se encuentra en PetStore. La URL del punto de enlace HTTP del backend es `http://petstore-demo-endpoint.execute-api.com/petstore/pets/`*n*, donde n es un entero como identificador de la mascota consultada.

La API admite el acceso a CORS a través de los métodos `OPTIONS` del tipo de integración `MOCK`. API Gateway devuelve los encabezados solicitados que admiten el acceso a CORS.

El siguiente procedimiento le guiará por los pasos para crear y probar una API a partir de un ejemplo mediante la consola de API Gateway.

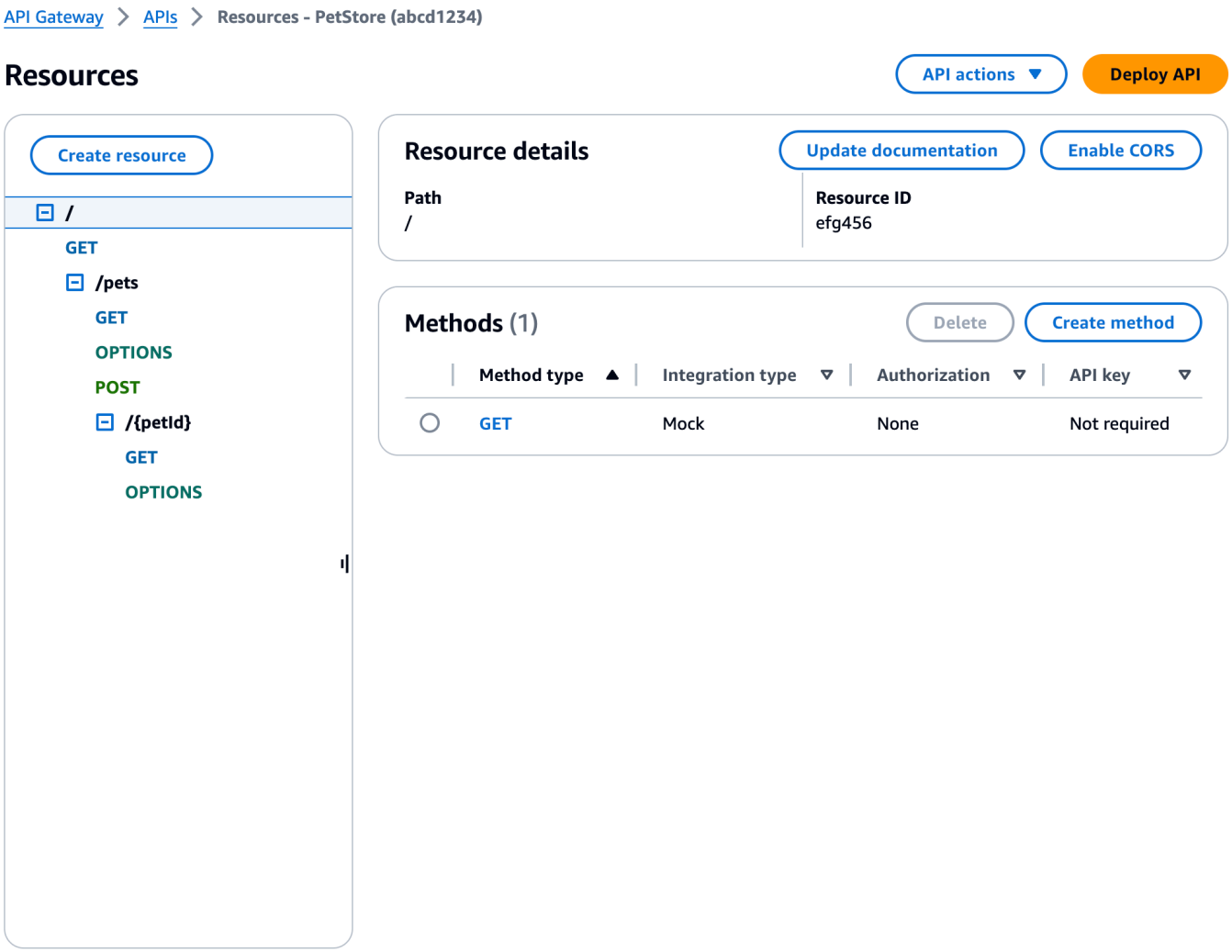
Para importar, desarrollar y probar la API de ejemplo

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Realice una de las siguientes acciones:
   - Para crear la primera API, para API de REST, elija Crear.
   - Si ha creado una API antes, elija Crear API y, a continuación, elija Crear para API de REST.
3. En Crear API de REST, elija API de ejemplo y, a continuación, elija Crear API para crear la API de ejemplo.



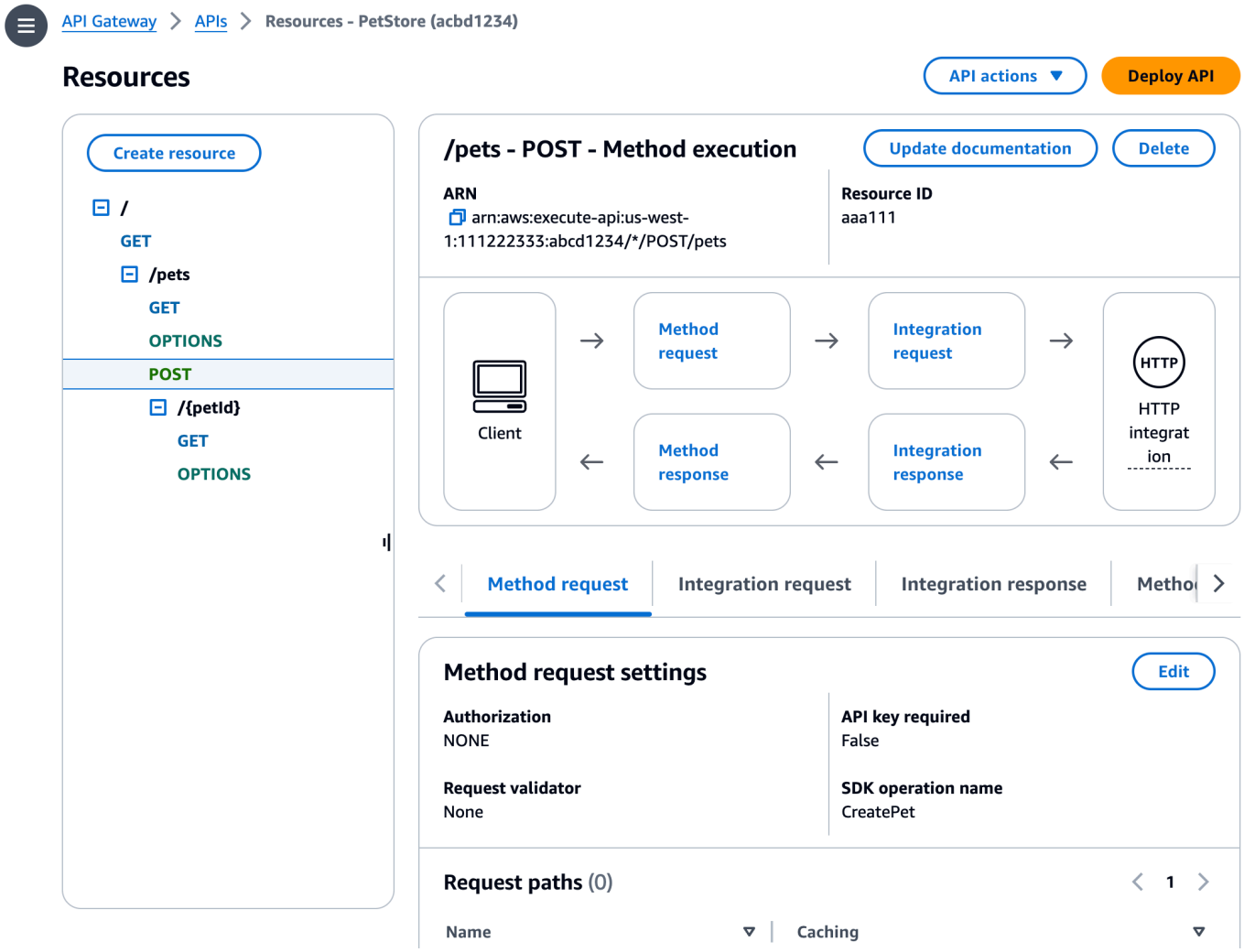
Puede desplazarse por la definición de OpenAPI para obtener información detallada sobre esta API de ejemplo antes de elegir Crear API.

4. En el panel de navegación principal, elija Recursos. La API recién creada se muestra de la siguiente forma:



El panel Resources (Recursos) muestra la estructura de la API creada como un árbol de nodos. Los métodos de API definidos en cada recurso son los extremos del árbol. Cuando se selecciona un recurso, todos sus métodos se muestran en el panel Métodos situado a la derecha. Junto a cada método se muestran el tipo de método, el tipo de integración, el tipo de autorización y el requisito de clave de API.

5. Para ver los detalles de un método, para modificar su configuración o para probar la invocación del método, elija el nombre del método en la lista de métodos o en el árbol de recursos. A continuación, elegimos el `POST /pets` método como ejemplo:



El panel de Ejecución de método resultante presenta una vista lógica de la estructura y el comportamiento del método elegido (`POST /pets`).

La Solicitud de método y la Respuesta de método representan la interfaz de la API con el frontend, y la Solicitud de integración y la Respuesta de integración representan la interfaz de la API con el backend.

El cliente utiliza la API para obtener acceso a una característica del backend a través de Solicitud de método. API Gateway traduce la solicitud del cliente, si fuera necesario, a un formato aceptable por el backend en Solicitud de integración antes de reenviar la solicitud entrante al backend. La solicitud transformada se conoce como la solicitud de integración. Del mismo modo, el backend devuelve la respuesta a API Gateway en Respuesta de integración. A continuación, API Gateway la dirige a Method Response (Respuesta de método) antes de

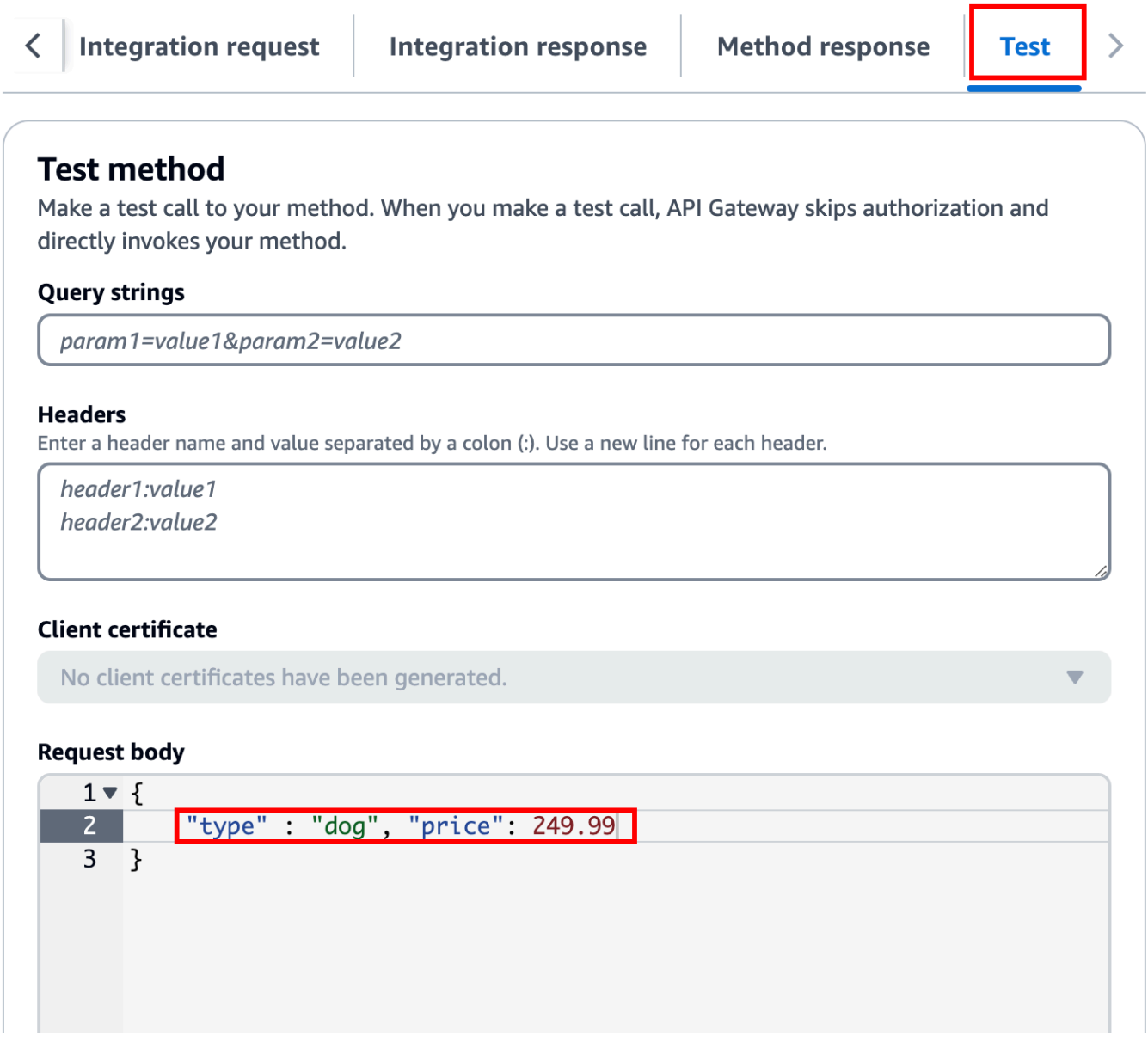
enviarla al cliente. De nuevo, si fuera necesario, API Gateway puede asignar los datos de la respuesta del backend a un formulario previsto por el cliente.

En el caso del método `POST` de recurso de API, la carga de la solicitud del método puede transmitirse a través de la solicitud de integración sin modificación si la carga de la solicitud de método está en el mismo formato que la carga de la solicitud de integración.

La solicitud del método `GET /` usa el tipo de integración `MOCK` y no está vinculada a ningún punto de enlace de backend real. La Respuesta de integración correspondiente está configurada para devolver una página HTML estática. Cuando se llama al método, API Gateway simplemente acepta la solicitud e inmediatamente devuelve la respuesta a la integración configurada al cliente a través de Respuesta de método. Puede utilizar la integración simulada para probar una API sin requerir un punto de enlace del backend. También puede utilizarla para servir una respuesta local generada partir de una plantilla de asignación de cuerpo de respuesta.

Como desarrollador de la API, puede controlar los comportamientos de las interacciones del frontend de la API mediante la configuración de la solicitud de método y una respuesta de método. Puede controlar los comportamientos de las interacciones del backend de la API mediante la configuración de la solicitud de integración y la respuesta de integración. Estos comportamientos implican asignaciones de datos entre un método y su integración correspondiente. Por el momento, nos centraremos en probar la API para proporcionar una experiencia de usuario completa.

6. Seleccione la pestaña Pruebas. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña.
7. Por ejemplo, para probar el método `POST /pets`, escriba la siguiente carga **`{"type": "dog","price": 249.99}`** en el Cuerpo de la solicitud antes de elegir el botón Pruebas.



La entrada especifica los atributos de la mascota que deseamos añadir a la lista de mascotas en el sitio web PetStore.

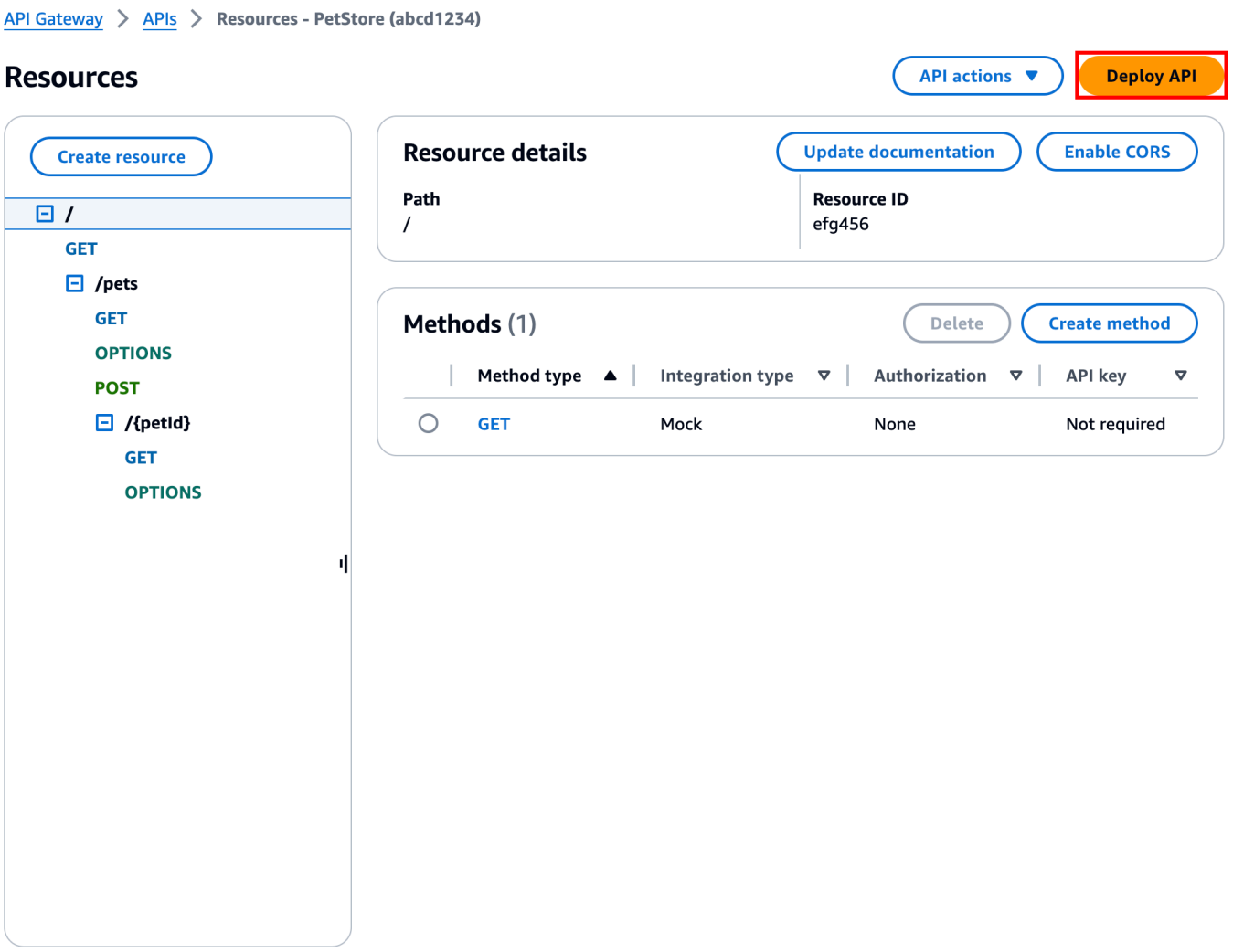
8. El resultado es el siguiente:



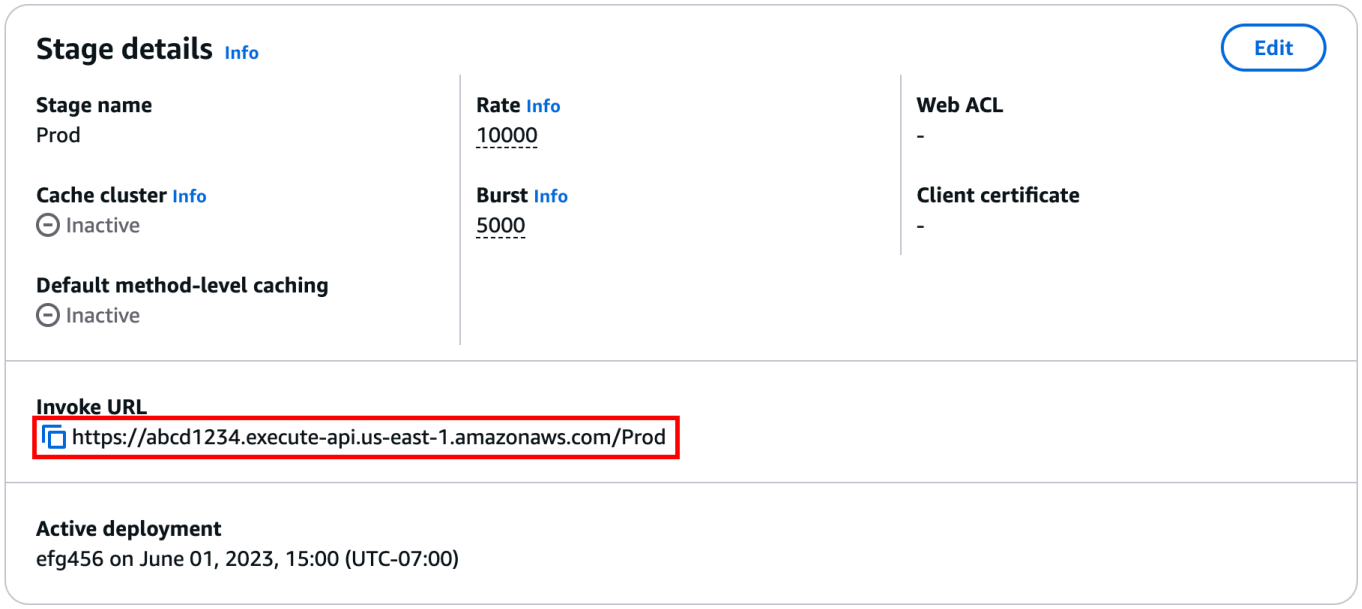
La entrada Registros de la salida muestra los cambios de estado de la solicitud del método a la solicitud de integración y de la respuesta de integración a la respuesta del método. Esto puede resultar útil para la resolución de errores de asignación que impidan que la solicitud se realice correctamente. En este ejemplo, la asignación no se aplica: la carga de la solicitud de método se transfiere a través de la solicitud integración al backend y, de forma parecida, la respuesta del backend se transfiere a través de la respuesta de integración al método de respuesta.

Para probar la API con un cliente distinto de la característica test-invoke-request de API Gateway, primero debe implementar la API en una etapa.

9. Elija Implementar API para implementar la API de ejemplo.



10. En Etapa, seleccione Nueva etapa y, a continuación, ingrese **`test`**.
11. (Opcional) En Description (Descripción), introduzca una descripción.
12. Elija Implementar.
13. En el panel Etapas resultante, en Detalles de la etapa, la URL de invocación muestra la dirección URL para invocar la solicitud del método `GET /` de la API.



14. Elija el icono de copiar para copiar la URL de invocación de la API y, a continuación, ingrese la URL de invocación de la API en un navegador web. Una respuesta correcta devuelve el resultado, generado a partir de la plantilla de asignación de la respuesta de integración.
15. En el panel de navegación Etapas, expanda la etapa prueba, seleccione GET en `/pets/{petId}` y, a continuación, copie el valor URL de invocación de `https://`*`api-id`*`.execute-api.`*`region`*`.amazonaws.com/test/pets/{petId}`. `{petId}` hace referencia a una variable de ruta.

    Pegue el valor de Invoke URL (URL de invocación) (obtenido en el paso anterior) en la barra de direcciones de un navegador, sustituyendo `{petId}` por, por ejemplo, `1` y, a continuación, pulse Intro para enviar la solicitud. Debería devolverse una respuesta 200 OK con la siguiente carga JSON:

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

    La invocación del método de la API tal como se muestra es posible porque su tipo Authorization (Autorización) está establecido en `NONE`. Si se ha utilizado la autorización de `AWS_IAM`, firmaría la solicitud con los protocolos Signature Version 4 (SigV4) o Signature Version 4a (SigV4a). Para ver un ejemplo de una solicitud de este tipo, consulte the section called “Tutorial: Creación de una API de REST con integración no de proxy HTTP”.

Elección de un tutorial de integración de HTTP

Para desarrollar una API con integración HTTP, puede utilizar una integración de proxy HTTP o una integración HTTP personalizada.

En una integración de proxy HTTP, solo tiene que establecer el método HTTP y el URI del punto de conexión HTTP, según los requisitos de backend. Le recomendamos utilizar integración de proxy HTTP cada vez que sea posible, para aprovechar la configuración de API simplificada.

Es posible que quiera usar una integración HTTP personalizada si necesita transformar los datos de la solicitud del cliente para el backend o transformar los datos de respuesta del backend para el cliente.

Temas

- Tutorial: Creación de una API de REST con integración de proxy HTTP
- Tutorial: Creación de una API de REST con integración no de proxy HTTP

Tutorial: Creación de una API de REST con integración de proxy HTTP

La integración de proxy HTTP es un mecanismo sencillo, potente y versátil para desarrollar una API que permita que una aplicación web obtenga acceso a múltiples recursos o características del punto de enlace HTTP integrado, por ejemplo, el sitio web completo, con una configuración simplificada de un único método de API. En la integración de proxy HTTP, API Gateway transmite la solicitud de método enviada por el cliente al backend. Los datos de la solicitud transmitida incluyen los encabezados de solicitud, los parámetros de cadena de consulta, las variables de ruta de la URL y la carga. El punto de enlace de HTTP del backend o del servidor web analiza los datos de la solicitud entrante para determinar la respuesta que devuelve. La integración de proxy HTTP permite que el cliente y el backend interactúen directamente sin intervención de API Gateway después de haber configurado el método de API, salvo cuando se producen problemas conocidos, como caracteres no admitidos, que se indican en the section called “Notas importantes”.

Con el amplio recurso de proxy `{proxy+}` y el verbo catch-all `ANY` para el método HTTP, puede utilizar una integración de proxy HTTP para crear una API de un único método de API. El método expone el conjunto completo de recursos y operaciones HTTP de acceso público de un sitio web. Cuando el servidor web del backend abre más recursos para acceso público, el cliente puede utilizar estos nuevos recursos con la misma configuración de API. Para habilitar esto, el desarrollador del sitio web debe comunicar con claridad al desarrollador del cliente cuáles son los nuevos recursos y cuáles son las operaciones aplicables para cada uno de ellos.

Como una introducción rápida, el siguiente tutorial demuestra la integración de proxy HTTP. En el tutorial creamos una API que utiliza la consola de API Gateway para integrarse con el sitio web PetStore mediante un recurso de proxy genérico `{proxy+}` y creamos el marcador de posición del método HTTP de `ANY`.

Temas

- Creación de una API con integración de proxy HTTP con la consola de API Gateway
- Probar una API con la integración de proxy HTTP

Creación de una API con integración de proxy HTTP con la consola de API Gateway

El siguiente procedimiento le guiará por los pasos para crear y probar una API con un recurso de proxy para un backend HTTP mediante la consola de API Gateway. El backend HTTP es el `PetStore` sitio web (`http://petstore-demo-endpoint.execute-api.com/petstore/pets`) de Tutorial: Creación de una API de REST con integración no de proxy HTTP, en el que las capturas de pantalla se utilizan como ayudas visuales para ilustrar los elementos de IU de API Gateway. Si es la primera vez que utiliza la consola de API Gateway para crear una API, le recomendamos seguir esa sección en primer lugar.

Para crear un API

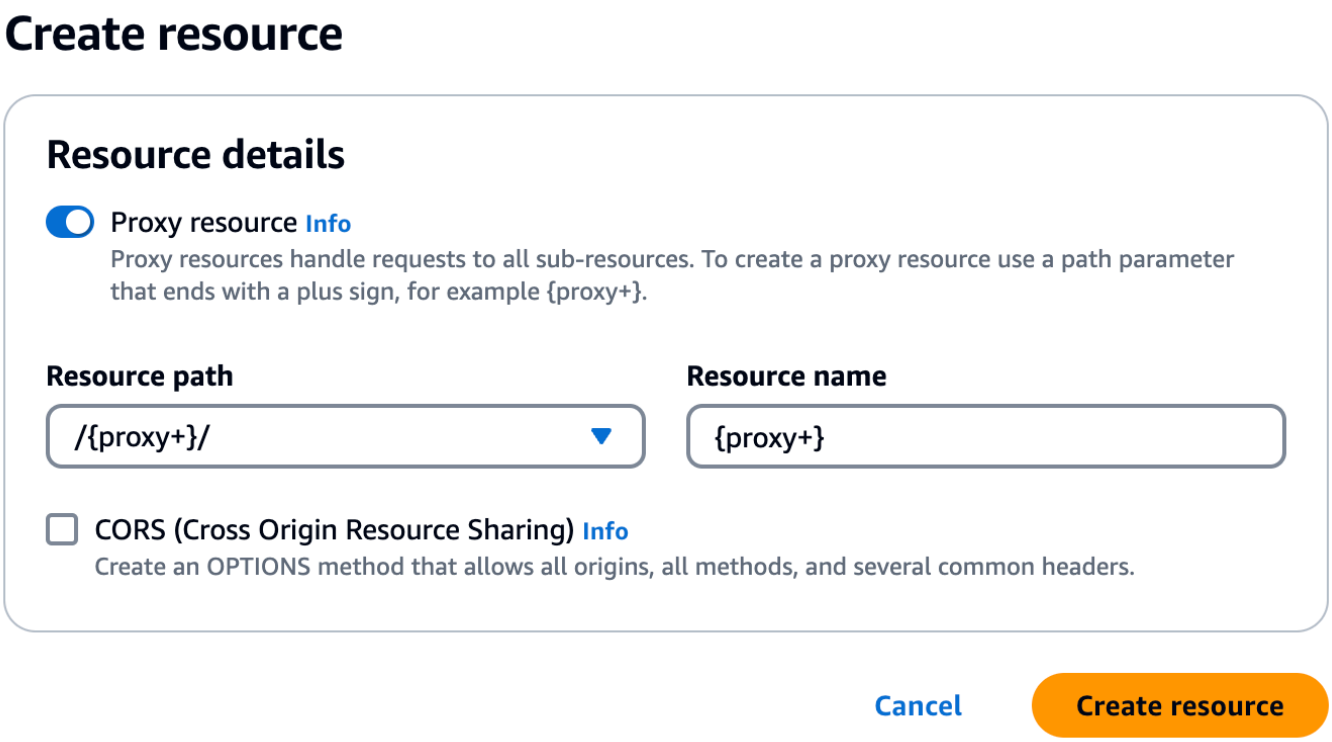
1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Si es la primera vez que utiliza API Gateway, verá una página en la que aparecen las características del servicio. En REST API, elija Build (Compilación). Cuando aparezca el menú emergente Create Example API (Crear API de ejemplo), elija OK (Aceptar).

   Si esta no es la primera vez que utiliza API Gateway, elija Create API (Crear API). En REST API, elija Build (Compilación).
3. En API name (Nombre de la API), escriba**`HTTPProxyAPI`**.
4. (Opcional) En Description (Descripción), introduzca una descripción.
5. Mantenga Tipo de punto de conexión de la API establecido en Regional.
6. En Tipo de dirección IP, seleccione IPv4.
7. Seleccione Crear API.

En este paso, creará una ruta de recursos de proxy de `{proxy+}`. Este es el marcador de posición de cualquier punto de conexión del backend bajo `http://petstore-demo-endpoint.execute-api.com/`. Por ejemplo, puede ser `petstore`, `petstore/pets` y `petstore/pets/{petId}`. API Gateway crea el método ANY al crear el recurso `{proxy+}` y actúa como un marcador de posición para cualquiera de los verbos HTTP admitidos en tiempo de ejecución.

Para crear un recurso /{proxy+}

1. Elija la API.
2. En el panel de navegación principal, elija Recursos.
3. Elija Crear recurso.
4. Active Recurso proxy.
5. Mantenga Ruta del recurso en `/`.
6. En Nombre del recurso, escriba **`{proxy+}`**.
7. Mantenga desactivado CORS (uso compartido de recursos entre orígenes).
8. Elija Crear recurso.



En este paso, integrará el método ANY con un punto de conexión HTTP de backend mediante una integración de proxy. En la integración de proxy, API Gateway transmite la solicitud de método enviada por el cliente al backend sin la intervención de API Gateway.

Para crear un método **ANY**

1. Elija el recurso /{proxy+}.
2. Elija el método ANY.
3. Bajo el símbolo de advertencia, seleccione Editar integración. No puede implementar una API que tenga un método sin una integración.
4. En Tipo de integración, seleccione HTTP.
5. Active Integración de proxy HTTP.
6. En Método HTTP, seleccione ANY.
7. En URL del punto de conexión, introduzca **`http://petstore-demo-endpoint.execute-api.com/{proxy}`**.
8. Seleccione Save.

Probar una API con la integración de proxy HTTP

El éxito de una solicitud de cliente específica dependerá de lo siguiente:

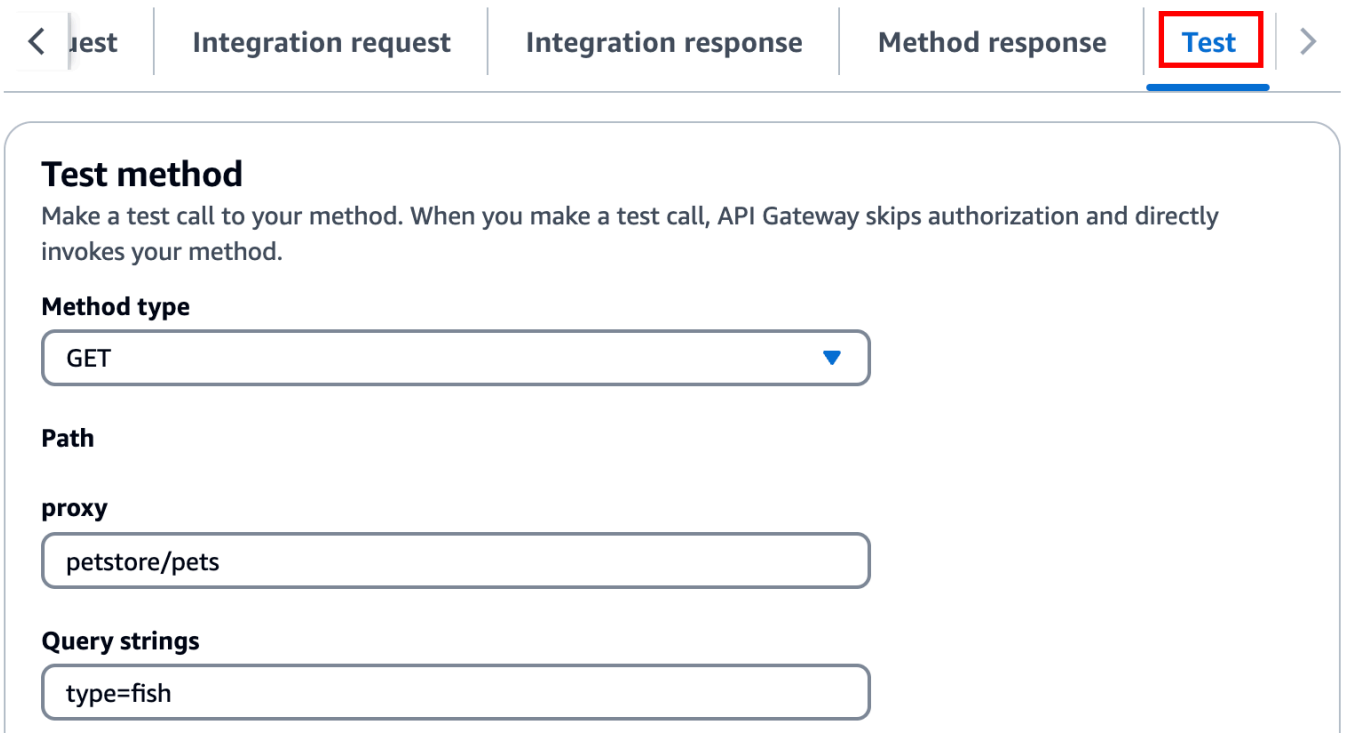
- Si el backend ha puesto a disposición el punto de enlace del backend correspondiente y, si lo ha hecho, si ha concedido los permisos de acceso requeridos.
- Si el cliente proporciona la entrada correcta.

Por ejemplo, la API de PetStore aquí utilizada no expone el recurso `/petstore`. Por lo tanto, obtendrá una respuesta `404 Resource Not Found` que contiene el mensaje de error de `Cannot GET /petstore`.

Además, el cliente debe tener la posibilidad de gestionar el formato de salida del backend para analizar los resultados correctamente. API Gateway no realiza mediaciones para facilitar las interacciones entre el cliente y el backend.

Para probar una API integrada con el sitio web PetStore utilizando la integración de proxy HTTP a través del recurso de proxy

1. Seleccione la pestaña Pruebas. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña.
2. En Tipo de método, seleccione `GET`.
3. En Ruta, debajo de proxy, introduzca **`petstore/pets`**.
4. En Cadenas de consulta, escriba **`type=fish`**.
5. Seleccione Probar.



Como que el sitio web del backend admite la solicitud `GET /petstore/pets?type=fish`, devuelve una respuesta correcta similar a la siguiente:

```
[
  {
    "id": 1,
    "type": "fish",
    "price": 249.99
  },
  {
    "id": 2,
```

```
    "type": "fish",
    "price": 124.99
  },
  {
    "id": 3,
    "type": "fish",
    "price": 0.99
  }
]
```

Si intenta llamar a `GET /petstore`, obtendrá una respuesta `404` con un mensaje de error `Cannot GET /petstore`. Esto se debe a que el backend no es compatible con la operación especificada. Si llama a `GET /petstore/pets/1`, obtendrá una respuesta `200 OK` con la siguiente carga, ya que la solicitud admite el sitio web PetStore.

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

También puede utilizar un navegador para probar la API. Implemente su API y asóciela a una etapa para crear la URL de invocación de su API.

Para implementar su API

1. Elija Implementar API.
2. En Etapa, seleccione Nueva etapa.
3. En Stage name (Nombre de etapa), escriba **`test`**.
4. (Opcional) En Description (Descripción), introduzca una descripción.
5. Elija Implementar.

Ahora los clientes pueden llamar a la API.

Para invocar la API

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija la API.

3. En el panel de navegación principal, elija Etapa.
4. En Detalles de la etapa, elija el icono de copia para copiar la URL de invocación de la API.

   Introduzca la URL de invocación de la API en un navegador web.

   La URL completa debería ser `https://abcdef123.execute-api.us-east-2.amazonaws.com/test/petstore/pets?type=fish`.

   Su navegador envía una `GET` solicitud a la API.
5. El resultado debe ser el mismo que el que se devuelve cuando utiliza Prueba desde la consola de API Gateway.

Tutorial: Creación de una API de REST con integración no de proxy HTTP

En este tutorial creará una API desde cero utilizando la consola de Amazon API Gateway. La consola se puede considerar como un estudio de diseño de la API y se puede utilizar para definir el ámbito de las características de API, para probar su comportamiento, para crear la API y para implementar la API en etapas.

Temas

- Crear una API con integración de HTTP personalizada
- (Opcional) Parámetros de la solicitud de mapeo

Crear una API con integración de HTTP personalizada

Esta sección le guía por los pasos para crear recursos, exponer métodos en un recurso, configurar un método para lograr los comportamientos de la API deseados y probar e implementar la API.

En este paso, se crea una API vacía. En los siguientes pasos, debe crear recursos y métodos para conectar la API al punto de conexión `http://petstore-demo-endpoint.execute-api.com/petstore/pets` mediante una integración HTTP sin proxy.

Para crear un API

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Si es la primera vez que utiliza API Gateway, verá una página en la que aparecen las características del servicio. En REST API, elija Build (Compilación). Cuando aparezca el menú emergente Create Example API (Crear API de ejemplo), elija OK (Aceptar).

Si esta no es la primera vez que utiliza API Gateway, elija Create API (Crear API). En REST API, elija Build (Compilación).

3. En API name (Nombre de la API), escriba**`HTTPNonProxyAPI`**.
4. (Opcional) En Description (Descripción), introduzca una descripción.
5. Mantenga Tipo de punto de conexión de la API establecido en Regional.
6. En Tipo de dirección IP, seleccione IPv4.
7. Seleccione Crear API.

El árbol Resources (Recursos) muestra el recurso raíz (`/`) sin métodos. En este ejercicio, vamos a desarrollar la API con la integración HTTP personalizada del sitio web de PetStore (http://petstore-demo-endpoint.execute-api.com/petstore/pets). Con fines ilustrativos, crearemos un recurso `/pets` como un elemento secundario de la raíz y expondremos un método GET en este recurso para que un cliente pueda recuperar una lista de los elementos Pets disponibles en el sitio web de PetStore.

Para crear un recurso /pets

1. Elija Crear recurso.
2. Mantenga Recurso proxy desactivado.
3. Mantenga Ruta del recurso en `/`.
4. En Nombre del recurso, escriba **`pets`**.
5. Mantenga desactivado CORS (uso compartido de recursos entre orígenes).
6. Elija Crear recurso.

En este paso, se crea un método `GET` en el recurso /pets. El método `GET` está integrado con el sitio web `http://petstore-demo-endpoint.execute-api.com/petstore/pets`. Otras opciones para un método de API incluyen las siguientes:

- POST, que se utiliza principalmente para crear recursos secundarios.
- PUT, que se utiliza principalmente para actualizar los recursos existentes (y, aunque no es recomendable, puede utilizarse para crear recursos secundarios).
- DELETE, que se utiliza para eliminar recursos.
- PATCH, que se utiliza para actualizar los recursos.

- HEAD, que se utiliza principalmente en escenarios de pruebas. Es igual que GET, pero no devuelve la representación de los recursos.
- OPTIONS, que pueden usar los intermediarios para obtener la información sobre las opciones de comunicación disponibles para el servicio de destino.

En el caso del HTTP method (Método HTTP) de la solicitud de integración, debe elegir uno de los métodos compatibles con el backend. Para `HTTP` o `Mock integration`, es razonable que la solicitud del método y la solicitud de integración usen el mismo verbo HTTP. Para otros tipos de integración, la solicitud del método usará probablemente un verbo HTTP diferente del de la solicitud de integración. Por ejemplo, para llamar a una función de Lambda, la solicitud de integración debe utilizar `POST` para invocar la función, mientras que la solicitud del método puede utilizar cualquier verbo HTTP en función de la lógica de la función de Lambda.

Para crear un método **`GET`** en el recurso /pets

1. Seleccione el recurso /pets.
2. Elija Crear método.
3. En Tipo de método, seleccione GET.
4. En Tipo de integración, seleccione Integración HTTP.
5. Mantenga desactivada Integración de proxy HTTP.
6. En Método HTTP, seleccione GET.
7. En URL del punto de conexión, introduzca **`http://petstore-demo-endpoint.execute-api.com/petstore/pets`**.

   El sitio web de PetStore le permite recuperar una lista de elementos `Pet` por tipo de mascota, por ejemplo, "Perro" o "Gato", en una determinada página.
8. En Tratamiento de contenido, seleccione Acceso directo.
9. Elija Parámetros de cadenas de consulta de URL.

   El sitio web de PetStore utiliza los parámetros de cadena de consulta `type` y `page` para aceptar una entrada. Se agregan los parámetros de cadena de consulta a la solicitud del método y se asignan a los parámetros de cadena de consulta correspondientes de la solicitud de integración.
10. Para agregar parámetros de cadena de consulta, haga lo siguiente:

    a. Elija Añadir cadena de consulta.

    b. En Nombre, introduzca **`type`**.

c. Mantenga desactivados Obligatorio y Almacenamiento en caché.

Repita los pasos anteriores para crear una cadena de consulta adicional con el nombre **`page`**.

11. Elija Crear método.

El cliente ahora puede proporcionar un tipo de mascota y un número de página como parámetros de cadena de consulta cuando envíe una solicitud. Estos parámetros de entrada deben mapearse a los parámetros de cadena de consulta de la integración para reenviar los valores de entrada a nuestro sitio web de PetStore en el backend.

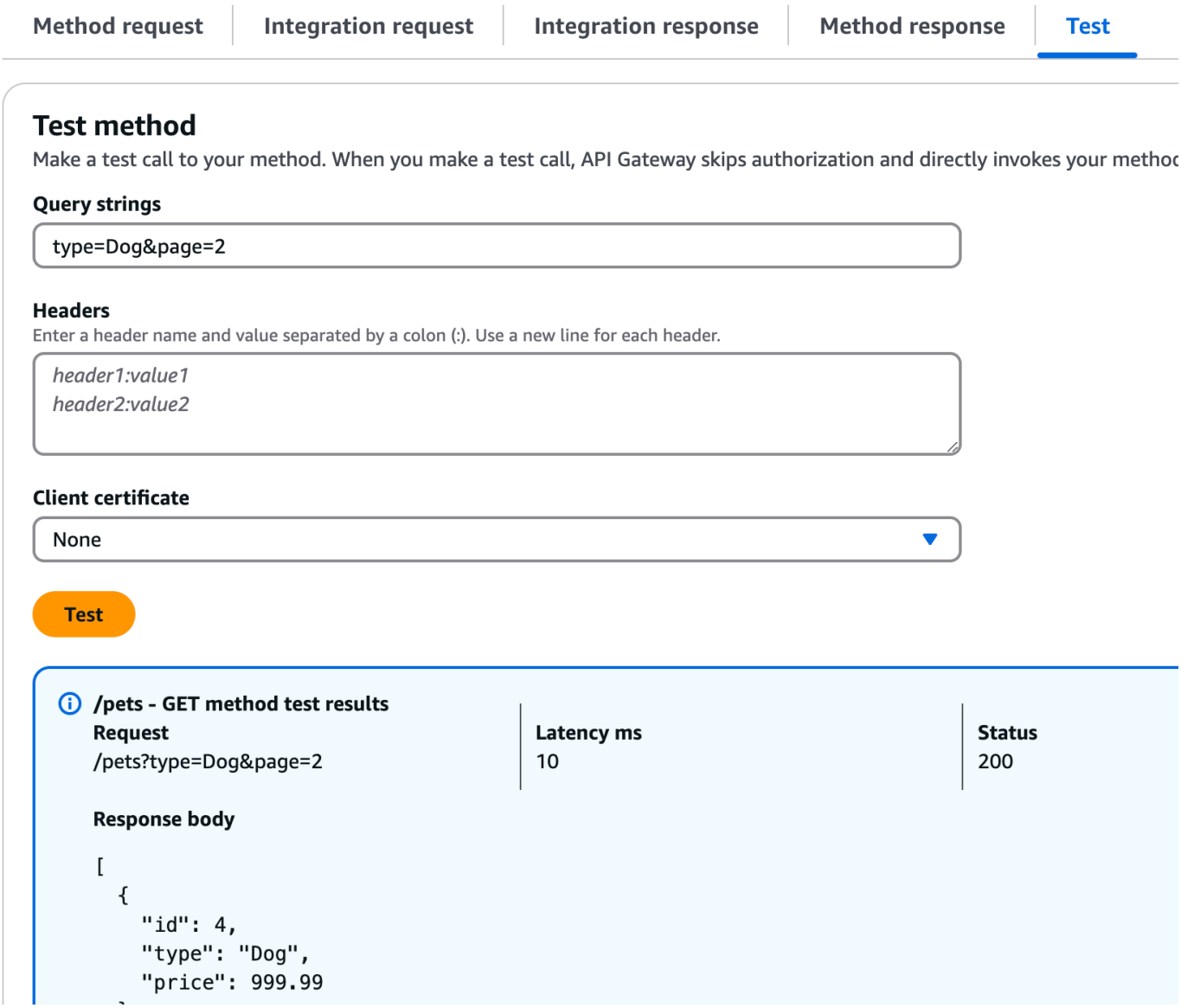
Para mapear los parámetros de entrada a la solicitud de integración

1. En la pestaña Solicitud de integración, en Configuración de la solicitud de integración, seleccione Editar.
2. Elija Parámetros de cadenas de consulta de URL y luego haga lo siguiente:
   a. Seleccione Añadir parámetro de cadena de consulta.
   b. En Nombre, escriba **`type`**.
   c. En Mapeado de, introduzca **`method.request.querystring.type`**.
   d. Mantenga Almacenamiento en caché desactivado.
   e. Seleccione Añadir parámetro de cadena de consulta.
   f. En Nombre, escriba **`page`**.
   g. En Mapeado de, introduzca **`method.request.querystring.page`**.
   h. Mantenga Almacenamiento en caché desactivado.
3. Seleccione Save.

Para probar el API

1. Elija la pestaña Prueba. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña.
2. En Cadenas de consulta, escriba **`type=Dog&page=2`**.
3. Seleccione Probar

   El resultado es similar al siguiente:



Ahora que la prueba se ha realizado correctamente, podemos implementar la API para ponerla a disposición del público en general.

4. Elija Implementar API.
5. En Etapa, seleccione Nueva etapa.
6. En Stage name (Nombre de etapa), escriba **`Prod`**.
7. (Opcional) En Description (Descripción), introduzca una descripción.
8. Elija Implementar.
9. (Opcional) En Detalles de la etapa, para URL de invocación, puede elegir el icono de copia para copiar la URL de invocación de su API. Puede utilizar esto con herramientas como Postman y cURL para probar la API.

Si utiliza un SDK para crear un cliente, puede llamar a los métodos expuestos por el SDK para firmar la solicitud. Para obtener más información, consulte los AWSSDK que desee.

> **Note**
>
> Cuando se realicen cambios en la API, deberá volver a implementar la API para que las características nuevas o actualizadas estén disponibles antes de volver a invocar la URL de la solicitud.

(Opcional) Parámetros de la solicitud de mapeo

Asignación de parámetros de solicitudes para una API de API Gateway

En este tutorial, se muestra cómo crear un parámetro de ruta `{petId}` en la solicitud de método de la API para especificar un ID de elemento, asignarlo al parámetro de ruta `{id}` en la URL de la solicitud de integración y enviar la solicitud al punto de conexión HTTP.

> **Note**
>
> Si introduce una letra con las mayúsculas o minúsculas incorrectas (por ejemplo, una letra minúscula en lugar de una mayúscula), se producirán errores más adelante en el tutorial.

Paso 1: Crear recursos

En este paso, se crea un recurso con un parámetro de ruta {petId}.

Para crear el recurso {petID}

1. Seleccione el recurso /pets y, a continuación, elija Crear recurso.
2. Mantenga Recurso proxy desactivado.
3. En Ruta de recurso, seleccione /pets/.
4. En Nombre del recurso, escriba **`{petId}`**.

   Utilice llaves (`{ }`) alrededor de `petId` para que aparezca /pets/{petId}.
5. Mantenga desactivado CORS (uso compartido de recursos entre orígenes).
6. Elija Crear recurso.

Paso 2: Crear y probar los métodos

En este paso, se crea un método `GET` con un parámetro de ruta `{petId}`.

Para configurar el método GET

1. Seleccione el recurso /{petId} y, a continuación, elija Crear método.
2. En Tipo de método, seleccione GET.
3. En Tipo de integración, seleccione Integración HTTP.
4. Mantenga desactivada Integración de proxy HTTP.
5. En Método HTTP, seleccione GET.
6. En URL del punto de conexión, introduzca **`http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}`**
7. En Tratamiento de contenido, seleccione Acceso directo.
8. Mantenga activado Tiempo de espera predeterminado.
9. Elija Crear método.

Ahora, asigne el parámetro de ruta `{petId}` que acaba de crear al parámetro de ruta `{id}` en la URL del punto de conexión HTTP de la solicitud de integración. La URL del punto de conexión HTTP era **`http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}`**.

Para mapear el parámetro de ruta **`{petId}`**

1. En la pestaña Solicitud de integración, en Configuración de la solicitud de integración, seleccione Editar.
2. Elija los Parámetros de la ruta URL.
3. API Gateway crea un parámetro de ruta para la solicitud de integración denominado petId. Sin embargo, este parámetro de ruta no es válido para la URL del punto de conexión HTTP que ha configurado como integración de backend. El punto de conexión HTTP utiliza `{id}` como parámetro de ruta. En Nombre, elimine petId y escriba **`id`**.

   Esto asigna el parámetro de ruta de la solicitud del método de `petId` al parámetro de ruta de la solicitud de integración de `id`.
4. Seleccione Save.

Ahora pruebe el método.

Para probar el método

1. Elija la pestaña Prueba. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña.
2. En Ruta, en petId, introduzca **4**.
3. Seleccione Test (Probar).

   Si todo sale bien, en Cuerpo de la respuesta se mostrará lo siguiente:

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

Paso 3: Implementar la API

En este paso, implementará la API para que pueda empezar a llamarla fuera de la consola de API Gateway.

Para implementar la API

1. Elija Implementar API.
2. En Etapa, seleccione Prod.
3. (Opcional) En Description (Descripción), introduzca una descripción.
4. Elija Deploy (Implementar).

Paso 4: Probar la API

En este paso, saldrá de la consola de API Gateway y utilizará la API para obtener acceso al punto de enlace HTTP.

1. En el panel de navegación principal, elija Etapa.
2. En Detalles de la etapa, elija el icono de copia para copiar la URL de invocación de la API.

   Debe tener un aspecto similar al siguiente:

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

3. Introduzca esta URL en el cuadro de dirección de una nueva pestaña del navegador y añada `/pets/4` a la URL antes de enviar su solicitud.
4. El navegador devolverá lo siguiente:

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

Siguientes pasos

Puede personalizar aún más su API activando la validación de solicitudes, transformando los datos o creando respuestas de puerta de enlace personalizadas.

Para explorar más formas de personalizar su API, consulte los siguientes tutoriales:

- Para obtener más información sobre la validación de solicitud, consulte Configuración de la validación básica de solicitudes en API Gateway.
- Para obtener información sobre cómo transformar las cargas de solicitudes y respuestas, consulte Tutorial: modificación de la solicitud y respuesta de integración para integraciones con servicios de AWS.
- Para obtener información sobre cómo crear respuestas de puerta de enlace personalizadas consulte Configurar una respuesta de gateway para una API REST mediante la consola de API Gateway.

Tutorial: Creación de una API de REST con una integración privada

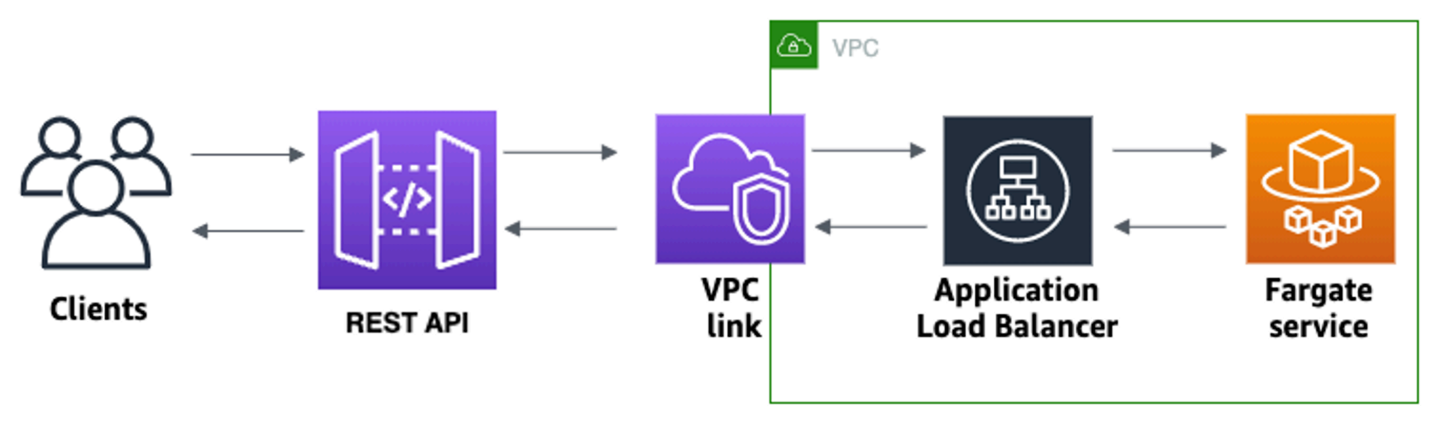
En este tutorial, se crea una API de REST que se conecta a un servicio de Amazon ECS que se ejecuta en una Amazon VPC. Los clientes fuera de su VPC de Amazon pueden utilizar la API para acceder al Servicio Amazon ECS.

Para completar este tutorial se necesita aproximadamente una hora. En primer lugar, utilice una plantilla de CloudFormation para crear una nube de Amazon VPC y un servicio Amazon ECS. Luego, utilice la consola de API Gateway para crear un enlace de VPC V2. El vínculo VPC permite a API Gateway acceder al Servicio Amazon ECS que se ejecuta en su Amazon VPC A continuación, se

crea una API de REST que utilice el enlace de VPC V2 para conectarse al servicio de Amazon ECS. Por último, se prueba la API.

Al invocar la API de REST, API Gateway envía la solicitud al servicio de Amazon ECS a través del enlace de VPC V2 y, a continuación, devuelve la respuesta del servicio.

> **Note**
>
> Anteriormente, este tutorial era compatible con las API HTTP y ahora lo es con las API de REST que utilizan el enlace de VPC V2.



Para completar este tutorial, necesita una cuenta de AWS y un usuario de AWS Identity and Access Management con acceso a la consola. Para obtener más información, consulte Requisitos previos.

Temas

- Paso 1: crear un Servicio Amazon ECS
- Paso 2: crear un enlace de VPC
- Paso 3: Crear una API de REST
- Paso 4: probar la API
- Paso 5: implemente su API
- Paso 6: llamar a la API
- Paso 7: limpiar

Paso 1: crear un Servicio Amazon ECS

Amazon ECS es un servicio de administración de contenedores que le facilita la tarea de ejecutar, detener y administrar contenedores de Docker en un clúster. En este tutorial, ejecute el clúster en una infraestructura sin servidor administrada por Amazon ECS.

Descargue y descomprima esta plantilla de CloudFormation que crea todas las dependencias del servicio, incluida una nube de Amazon VPC. Utilice la plantilla para crear un Servicio Amazon ECS que utilice un Application Load Balancer.

Para crear una pila de CloudFormation

1. Abra la consola de CloudFormation en https://console.aws.amazon.com/cloudformation.
2. Seleccione Create stack (Crear pila) y, a continuación, seleccione With new resources (standard) (Con nuevos recursos [estándar]).
3. En Specify template (Especificar plantilla), elija Upload a template file (Cargar un archivo de plantilla).
4. Seleccione la plantilla que ha descargado.
5. Elija Next (Siguiente).
6. En Stack name (Nombre de pila), escriba **`rest-api-private-integrations-tutorial`** y, a continuación, elija Next (Siguiente).
7. En Configure stack options (Configurar opciones de pila), elija Next (Siguiente).
8. Para Capabilities (Capacidades), sepa que CloudFormation puede crear recursos de IAM en su cuenta.
9. Elija Siguiente y, a continuación, elija Enviar.

CloudFormation aprovisiona al Servicio ECS. Puede tardar unos minutos. Cuando el estado de la pila de CloudFormation sea CREATE_COMPLETE, estará listo para continuar con el paso siguiente.

Paso 2: crear un enlace de VPC

Un enlace de VPC permite a API Gateway acceder a recursos privados en una Amazon VPC Utilice un enlace de VPC para permitir a los clientes acceder al servicio de Amazon ECS a través de la API de REST.

Para crear un enlace de VPC

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Enlaces de VPC y, a continuación, elija Crear.

   Es posible que necesite elegir el icono del menú para abrir el panel de navegación principal.
3. En Elegir una versión de enlace de VPC, seleccione Enlace de VPC V2.
4. En Name (Nombre) escriba **`private-integrations-tutorial`**.
5. En VPC, elija la VPC que ha creado en el paso 1. El nombre debe empezar por RestApiStack.
6. En Subnets (Subredes), seleccione las dos subredes privadas de la VPC. Los nombres terminan en `PrivateSubnet`.
7. Para Grupos de seguridad, seleccione el ID de grupo que comience por `private-integrations-tutorial` y tenga la descripción `RestApiStack/RestApiTutorialService/Service/SecurityGroup`.
8. Seleccione Crear.

Después de crear el enlace de VPC V2, API Gateway aprovisiona las interfaces de red elásticas para acceder a la VPC. El proceso puede demorar unos minutos. Mientras tanto, puede crear su API.

Paso 3: Crear una API de REST

La API de REST proporciona un punto de conexión HTTP para el servicio de Amazon ECS.

Para crear una API de REST

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Seleccione Create API (Crear API)y, a continuación, para API REST, seleccione Build (Construir).
3. En Nombre, escriba **`private-integration-api`**.
4. En Tipo de dirección IP, seleccione IPv4.
5. Seleccione Crear API.

   Después de crear la API, se crea un método.
6. Elija Crear método y luego haga lo siguiente:

   a. En Tipo de método, seleccione `GET`.

b. En Tipo de integración, seleccione Enlace de VPC.

c. Active Integración de proxy de VPC.

d. En Método HTTP, seleccione `GET`.

e. Para el enlace de la VPC, elija el enlace de VPC V2 que creó en el paso anterior.

f. En Objetivo de integración, ingrese el equilibrador de carga que creó con la plantilla de CloudFormation en el paso 1. Su nombre debe empezar por rest-.

g. En URL del punto de conexión, introduzca `http://private-integrations-tutorial.com`.

   La URL se utiliza para definir el encabezado `Host` de la solicitud de integración. En este caso, el encabezado del host es **`private-integrations-tutorial`**.

h. Elija Crear método.

   Con la integración del proxy, la API está lista para probarse.

Paso 4: probar la API

A continuación, se prueba a invocar el método de la API.

Para probar la API

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija la API.
3. Elija la pestaña Prueba. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña.
4. Seleccione Probar

   Compruebe que la respuesta de la API sea un mensaje de bienvenida que le indique que su aplicación se está ejecutando en Amazon ECS.

Paso 5: implemente su API

A continuación, debe implementar la API.

Para implementar su API

1. Elija Implementar API.

2. En Etapa, seleccione Nueva etapa.
3. En Stage name (Nombre de etapa), escriba **`Prod`**.
4. (Opcional) En Description (Descripción), introduzca una descripción.
5. Elija Implementar.

Paso 6: llamar a la API

Después de implementar la API, puede llamarla.

Llamada a la API

1. Ingrese la URL de invocación en un navegador web.

   La URL completa debería ser `https://abcd123.execute-api.us-east-2.amazonaws.com/Prod`.

   Su navegador envía una `GET` solicitud a la API.
2. Compruebe que la respuesta de la API sea un mensaje de bienvenida que le indique que su aplicación se está ejecutando en Amazon ECS.

   Si ve el mensaje de bienvenida, ha creado correctamente un servicio de Amazon ECS que se ejecuta en una Amazon VPC y ha utilizado una API de REST de API Gateway con un enlace de VPC V2 para acceder al servicio de Amazon ECS.

Paso 7: limpiar

Para evitar costos innecesarios, elimine los recursos creados como parte de este tutorial Los pasos siguientes eliminan el enlace de VPC V2, la pila de CloudFormation y la API de REST.

Para eliminar una API REST

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. En la página API, seleccione una API. Elija Action (Acciones), elija Delete (Eliminar)y, a continuación, confirme su elección.

Para eliminar un enlace de VPC

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.

2. Elija VPC link (Enlace de VPC).
3. Seleccione el enlace de VPC, elija Delete (Eliminar)y, a continuación, confirme su elección.

Para eliminar una pila de CloudFormation

1. Abra la consola de CloudFormation en https://console.aws.amazon.com/cloudformation.
2. Seleccione su pila de CloudFormation.
3. Elija Delete (Eliminar) y, a continuación, confirme su elección.

Tutorial: Creación de una API de REST con una integración de AWS

Tanto los temas Tutorial: Creación de una API de REST con una integración de proxy de Lambda como Tutorial: Creación de una API de REST con una integración no de proxy de Lambda describen cómo se debe crear una API de API Gateway para exponer la función de Lambda integrada. Además, puede crear una API de API Gateway para exponer otros servicios de AWS, como Amazon SNS, Amazon S3, Amazon Kinesis e incluso AWS Lambda. Esto es posible mediante la integración de `AWS`. La integración de Lambda o la integración de proxy de Lambda es un caso especial, donde la invocación de la función de Lambda se expone a través de la API de API Gateway.

Todos los servicios de AWS admiten API dedicadas para exponer sus características. Sin embargo, los protocolos de la aplicación o las interfaces de programación probablemente difieren entre un servicio y otro. Una API de API Gateway con la integración de `AWS` tiene la ventaja de proporcionar un protocolo de aplicación coherente para que su cliente tenga acceso a diferentes servicios de AWS.

En este tutorial, vamos a crear una API para exponer Amazon SNS. Para ver más ejemplos de integración de una API con otros servicios de AWS, consulte Tutoriales y talleres sobre Amazon API Gateway.

A diferencia de la integración de proxy de Lambda, no existe una integración de proxy correspondiente para otros servicios de AWS. Por lo tanto, un método de API se integra con una única acción de AWS. Para mayor flexibilidad, similar a la de la integración de proxy, puede configurar una integración de proxy de Lambda. Entonces, la función de Lambda analiza y procesa solicitudes para otras acciones de AWS.

API Gateway no vuelve a intentar las operaciones cuando se agota el tiempo de espera del punto de conexión. El intermediario de la API debe implementar una lógica de reintentos para administrar los tiempos de espera del punto de conexión.

Este tutorial se basa en las instrucciones y conceptos en Tutorial: Creación de una API de REST con una integración no de proxy de Lambda. Si aún no ha completado ese tutorial, le sugerimos que lo haga primero.

Temas

- Requisitos previos
- Paso 1: Crear el rol de ejecución del proxy de servicio de AWS
- Paso 2: Crear el recurso
- Paso 3: Crear el método GET
- Paso 4: Especificar la configuración del método y probar el método
- Paso 5: Implementar la API
- Paso 6: Probar la API
- Paso 7: Limpieza

Requisitos previos

Antes de empezar este tutorial, haga lo siguiente:

1. Realice los pasos que se indican en Configuración para usar API Gateway.
2. Cree una nueva API llamada `MyDemoAPI`. Para obtener más información, consulte Tutorial: Creación de una API de REST con integración no de proxy HTTP.
3. Implemente la API al menos una vez en una etapa denominada `test`. Para obtener más información, consulte Implementar la API en Elección de un tutorial de integración de AWS Lambda.
4. Realice el resto de los pasos de Elección de un tutorial de integración de AWS Lambda.
5. Cree al menos un tema en Amazon Simple Notification Service (Amazon SNS). Utilizará la API implementada para obtener una lista de temas de Amazon SNS asociados a su cuenta de AWS. Para obtener información sobre cómo crear un tema en Amazon SNS, consulte Creación de un tema. (No es necesario que copie el ARN del tema mencionado en el paso 5).

Paso 1: Crear el rol de ejecución del proxy de servicio de AWS

Para permitir que la API invoque acciones de Amazon SNS, debe tener las políticas de IAM adecuadas asociadas a un rol de IAM. En este paso, deberá crear un nuevo rol de IAM.

Para crear el rol de ejecución del proxy de servicio de AWS

1. Inicie sesión en la Consola de administración de AWS y abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. Elija Roles.
3. Elija Creación de rol.
4. Elija Servicio de AWS en Seleccionar el tipo de entidad de confianza y, a continuación, elija API Gateway y seleccione Permite que API Gateway envíe registros a CloudWatch Logs.
5. Seleccione Siguiente y de nuevo Siguiente.
6. En Nombre de rol, escriba **`APIGatewaySNSProxyPolicy`** y luego elija Crear rol.
7. En la lista Roles, elija el rol que acaba de crear. Puede que tenga que desplazarse o usar la barra de búsqueda para encontrar el rol.
8. Para el rol seleccionado, seleccione la pestaña Agregar permisos.
9. Elija Adjuntar políticas en la lista desplegable.
10. En la barra de búsqueda, escriba **`AmazonSNSReadOnlyAccess`** y, a continuación, elija Añadir permisos.

    > **Note**
    >
    > Para simplificar el proceso, este tutorial utiliza una política administrada. Como práctica recomendada, se deben crear políticas de IAM propias para otorgar los permisos mínimos requeridos.

11. Anote el ARN del rol recién creado, ya que lo usará más adelante.

Paso 2: Crear el recurso

En este paso, creará un recurso que permitirá al proxy de servicio de AWS interactuar con el servicio de AWS.

Para crear el recurso

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija la API.
3. Seleccione el recurso raíz, /, representado por una única barra oblicua (/), y después elija Crear recurso.

4. Mantenga Recurso proxy desactivado.
5. Mantenga Ruta del recurso en `/`.
6. En Nombre del recurso, escriba **`mydemoawsproxy`**.
7. Mantenga desactivado CORS (uso compartido de recursos entre orígenes).
8. Elija Crear recurso.

Paso 3: Crear el método GET

En este paso, creará un método GET que permitirá al proxy de servicio de AWS interactuar con el servicio de AWS.

Para crear el método **`GET`**

1. Seleccione el recurso /mydemoawsproxy y, a continuación, elija Crear método.
2. En el tipo de método, seleccione GET.
3. En Tipo de integración, seleccione Servicio de AWS.
4. En Región de AWS, seleccione la Región de AWS en la que ha creado el tema de Amazon SNS.
5. En Servicio de AWS, seleccione Amazon SNS.
6. Deje Subdominio de AWS en blanco.
7. En Método HTTP, seleccione GET.
8. En Tipo de acción, elija Usar nombre de acción.
9. En Nombre de la función, introduzca **`ListTopics`**.
10. En Rol de ejecución, escriba el ARN del rol para **`APIGatewaySNSProxyPolicy`**.
11. Elija Crear método.

Paso 4: Especificar la configuración del método y probar el método

Ahora puede probar el método `GET` para verificar que se ha configurado correctamente para enumerar los temas de Amazon SNS.

Para probar el método **`GET`**

1. Elija la pestaña Prueba. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña.

2. Seleccione Probar

   El resultado muestra una respuesta similar a la siguiente:

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

Paso 5: Implementar la API

En este paso, implementará la API para que pueda llamarla desde fuera de la consola de API Gateway.

Para implementar la API

1. Elija Implementar API.
2. En Etapa, seleccione Nueva etapa.
3. En Stage name (Nombre de etapa), escriba **`test`**.
4. (Opcional) En Description (Descripción), introduzca una descripción.
5. Elija Deploy (Implementar).

Paso 6: Probar la API

En este paso, salga de la consola de API Gateway y utilice el proxy de servicio de AWS para interactuar con el se rvicio de Amazon SNS.

1. En el panel de navegación principal, elija Etapa.
2. En Detalles de la etapa, elija el icono de copia para copiar la URL de invocación de la API.

   Debería tener un aspecto similar al siguiente:

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

3. Introduzca la URL en el cuadro de direcciones de una nueva pestaña del navegador.
4. Añada `/mydemoawsproxy` para que la URL tenga el siguiente aspecto:

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

   Desplácese hasta la dirección URL. Se debe mostrar información similar a la siguiente:

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

Paso 7: Limpieza

Puede eliminar los recursos de IAM que el proxy de servicio de AWS necesita para funcionar.

> **Warning**
>
> Si elimina un recurso de IAM que utiliza un proxy de servicio de AWS, ese proxy de servicio de AWS y todas las API que lo utilicen dejarán de funcionar. La eliminación de un recurso de IAM no se puede deshacer. Si desea utilizar el recurso de IAM de nuevo, debe volver a crearlo.

Para eliminar los recursos de IAM asociados

1. Abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el área Details (Detalles), elija Roles (Funciones).
3. Seleccione APIGatewayAWSProxyExecRole y, a continuación, elija Role Actions (Acciones de rol), Delete Role (Eliminar rol). Cuando se le pregunte, elija Yes, Delete.
4. En el área Details (Detalles), elija Policies (Políticas).
5. Seleccione APIGatewayAWSProxyExecPolicy y, a continuación, elija Policy Actions (Acciones de la política) y luego Delete (Eliminar). Cuando se le pregunte, elija Delete (Eliminar).

Ha llegado al final de este tutorial. Para obtener más detalles sobre la creación de una API con un proxy de servicio de AWS, consulte Tutorial: Creación de una API de REST como proxy de Amazon S3, Tutorial: Creación de una API de REST de calculadora con dos integraciones de servicios de AWS y una integración de Lambda sin proxy o Tutorial: Creación de una API de REST como proxy de Amazon Kinesis.

Tutorial: Creación de una API de REST de calculadora con dos integraciones de servicios de AWS y una integración de Lambda sin proxy

El the section called “Tutorial: Creación de una API de REST con una integración no de proxy de Lambda” solamente utiliza la integración de `Lambda Function`. La integración de `Lambda Function` es un caso especial del tipo de integración de `AWS Service` que configura por usted la mayor parte de la integración; por ejemplo, la incorporación automática de los permisos basados en recursos necesarios para invocar la función de Lambda. Aquí, dos de las tres integraciones usan integración de `AWS Service`. En este tipo de integración, tiene más control, pero tendrá que realizar manualmente tareas como la creación y especificación de un rol de IAM que contenga los permisos adecuados.

En este tutorial, creará una función `Calc` de Lambda que implementa operaciones aritméticas básicas, aceptando y devolviendo entradas y salidas con formato JSON. A continuación, creará una API REST y la integrará con la función de Lambda de la siguiente forma:

1. Exponiendo un método de `GET` en el recurso `/calc` para invocar la función de Lambda, proporcionando la entrada como parámetros de cadena de consulta. (Integración de `AWS Service`)

2. Exponiendo un método de `POST` en el recurso `/calc` para invocar la función de Lambda, proporcionando la entrada en la carga de la solicitud del método. (Integración de `AWS Service`)
3. Exponiendo un método `GET` en recursos `/calc/{operand1}/{operand2}/{operator}` anidados para invocar la función de Lambda, proporcionando la entrada como parámetros de ruta. (Integración de `Lambda Function`)

Además de probar con este tutorial, sugerimos que estudie el archivo de definición de OpenAPI para la API de `Calc`, que puede importar a API Gateway siguiendo las instrucciones de the section called "OpenAPI".

Temas

- Crear un rol de IAM asumible
- Creación de una función Calc de Lambda
- Probar la función Calc de Lambda
- Cree una API de Calc
- Integración 1: Crear un método GET con parámetros de consulta para llamar a la función de Lambda
- Integración 2: Crear un método POST con una carga JSON para llamar a la función de Lambda
- Integración 3: Crear un método GET con parámetros de ruta para llamar a la función de Lambda
- Definiciones de OpenAPI de una API de ejemplo integrada con una función de Lambda

Crear un rol de IAM asumible

Para que su API invoque su función `Calc` de Lambda, necesitará tener un rol de IAM asumible por API Gateway, que es un rol de IAM con la siguiente relación de confianza:

JSON

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

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

El rol que cree tendrá que tener el permiso InvokeFunction de Lambda. De lo contrario, el intermediario de la API recibirá una respuesta `500 Internal Server Error`. Para dar al rol este permiso, asociará la siguiente política de IAM:

JSON

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

A continuación, se explica cómo realizar todo esto:

Crear un rol de IAM asumible por API Gateway

1. Inicie sesión en la consola de IAM.
2. Elija Roles.
3. Elija Create Role.
4. En Seleccionar el tipo de entidad de confianza, elija Servicio de AWS.
5. En Choose the service that will use this role (Elegir el servicio que usará este rol), elija Lambda.
6. Elija Next: Permissions (Siguiente: permisos).
7. Elija Create Policy (Crear política).

   Se abrirá una nueva ventana de consola Create Policy (Crear política). En esa ventana, haga lo siguiente:

a. En la pestaña JSON, reemplace la política existente por la siguiente:

JSON

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

b. Elija Review policy (Revisar política).

c. En Review Policy (Revisar política) haga lo siguiente:

i. En Nombre, escriba un nombre, como **`lambda_execute`**.

ii. Elija Create Policy (Crear política).

8. En la ventana de consola Create Role (Crear rol) original, haga lo siguiente:

a. En Attach permissions policies (Asociar políticas de permisos), elija la política **`lambda_execute`** de la lista desplegable.

Si no ve su política en la lista, haga clic en el botón Refresh (Actualizar) en la parte superior de la lista. (¡No actualice la página del navegador!)

b. Elija Next: Tags (Siguiente: Etiquetas).

c. Elija Next: Review.

d. En Role name (Nombre de rol), escriba un nombre como **`lambda_invoke_function_assume_apigw_role`**.

e. Elija Create role (Crear rol).

9. Elija el rol **`lambda_invoke_function_assume_apigw_role`** de la lista de roles.

10. Seleccione la pestaña Trust Relationships (Relaciones de confianza).

11. Elija Edit trust relationship (Editar relación de confianza).

12. Reemplace la política existente por la siguiente:

    JSON

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

13. Elija Update Trust Policy.
14. Anote el ARN del rol que acaba de crear. Lo necesitará más adelante.

Creación de una función **`Calc`** de Lambda

A continuación, creará una función de Lambda con la consola de Lambda.

1. En la consola de Lambda, elija Create function (Crear función).
2. Elija Author from Scratch (Crear desde cero).
3. En Name (Nombre) escriba **`Calc`**.
4. En Tiempo de ejecución, elija el último tiempo de ejecución de Node.js o Python compatible.
5. Para todas las demás opciones, utilice la configuración predeterminada.
6. Seleccione Creación de función.
7. Copie la siguiente función de Lambda en el tiempo de ejecución de su preferencia y péguela en el editor de código en la consola de Lambda.

Node.js

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

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

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

```
  }

  return res;
};
```

Python

```
import json


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

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

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

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

    return res
```

8. Bajo Execution role (Rol de ejecución) elija Choose an existing role (Elegir un rol existente).

9. Escriba el ARN del rol para el rol **`lambda_invoke_function_assume_apigw_role`** que creó anteriormente.
10. Elija Deploy (Implementar).

Esta función requiere dos operandos (a y b) y un operador (op) del parámetro de entrada `event`. La entrada es un objeto JSON con el siguiente formato:

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

Esta función devuelve el resultado calculado (c) y la entrada. Si se trata de una entrada no válida, la función devuelve el valor null o la cadena "Invalid op" como resultado. La salida tiene el siguiente formato JSON:

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

Debe probar la función en la consola de Lambda antes de integrarla con la API en el siguiente paso.

Probar la función **`Calc`** de Lambda

Indicamos aquí como probar la función `Calc` en la consola de Lambda.

1. Elija la pestaña Prueba.
2. Para el nombre del evento de prueba, escriba **`calc2plus5`**.
3. Sustituya la definición del evento de prueba por lo siguiente:

```
{
```

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

4. Seleccione Save.
5. Seleccione Probar
6. Expanda Execution result: succeeded (Resultado de la ejecución: correcta). Debería ver lo siguiente:

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

Cree una API de **`Calc`**

El procedimiento siguiente describe cómo crear una API para la función `Calc` de Lambda que acaba de crear. En las siguientes secciones, añadirá recursos y métodos.

Para crear un API

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Si es la primera vez que utiliza API Gateway, verá una página en la que aparecen las características del servicio. En REST API, elija Build (Compilación). Cuando aparezca el menú emergente Create Example API (Crear API de ejemplo), elija OK (Aceptar).

   Si esta no es la primera vez que utiliza API Gateway, elija Create API (Crear API). En REST API, elija Build (Compilación).
3. En API name (Nombre de la API), escriba**`LambdaCalc`**.
4. (Opcional) En Description (Descripción), introduzca una descripción.
5. Mantenga Tipo de punto de conexión de la API establecido en Regional.
6. En Tipo de dirección IP, seleccione IPv4.
7. Seleccione Create API (Crear API).

Integración 1: Crear un método **GET** con parámetros de consulta para llamar a la función de Lambda

Mediante la creación de un método GET que pasa los parámetros de cadenas de consulta a la función de Lambda, habilita la API que se va a invocar desde un navegador. Este método puede ser útil, en especial para las API que permiten acceso abierto.

Después de crear una API, se crea un recurso. Normalmente, los recursos de la API están organizados en un árbol de recursos de acuerdo con la lógica de la aplicación. Para este paso, debe crear un recurso /calc.

Para crear un recurso /calc

1. Elija Crear recurso.
2. Mantenga Recurso proxy desactivado.
3. Mantenga Ruta del recurso en `/`.
4. En Nombre del recurso, escriba **`calc`**.
5. Mantenga desactivado CORS (uso compartido de recursos entre orígenes).
6. Elija Crear recurso.

Mediante la creación de un método GET que pasa los parámetros de cadenas de consulta a la función de Lambda, habilita la API que se va a invocar desde un navegador. Este método puede ser útil, en especial para las API que permiten acceso abierto.

En este método, Lambda requiere que se use la solicitud POST para invocar cualquier función de Lambda. Este ejemplo muestra que el método HTTP en una solicitud de método del frontend puede diferir de la solicitud de integración en el backend.

Para crear un método **GET**

1. Seleccione el recurso calc y, a continuación, elija Crear método.
2. En Tipo de método, seleccione GET.
3. En Tipo de integración, seleccione Servicio de AWS.
4. En Región de AWS, seleccione la Región de AWS donde creó la función de Lambda.
5. En Servicio de AWS, seleccione Lambda.
6. Deje Subdominio de AWS en blanco.

7. En Método HTTP, seleccione POST.
8. En Tipo de acción, elija Usar sustitución de ruta. Esta opción nos permite especificar el ARN de la acción Invoke para ejecutar nuestra función `Calc`.
9. En Sustitución de ruta, escriba **`2015-03-31/functions/arn:aws:lambda:`*`us-east-2`*`:`*`account-id`*`:function:Calc/invocations`**. En *`account-id`*, introduzca su identificador de Cuenta de AWS. En *`us-east-2`*, introduzca la Región de AWS donde creó la función de Lambda.
10. En Rol de ejecución, escriba el ARN del rol para **`lambda_invoke_function_assume_apigw_role`**.
11. No cambie la configuración de Caché de credenciales ni Tiempo de espera predeterminado.
12. Elija Configuración de solicitud de método.
13. En Validador de solicitud, seleccione Validar parámetros de cadena de consulta y encabezados.

    Esta configuración hará que se devuelva un mensaje de error si el cliente no especifica los parámetros requeridos.
14. Elija Parámetros de cadenas de consulta de URL.

    Ahora, debe configurar los parámetros de cadena de consulta para el método GET en el recurso /calc para que pueda recibir la entrada en nombre de la función de Lambda del backend.

    Para crear parámetros de cadena de consulta, haga lo siguiente:

    a. Elija Añadir cadena de consulta.
    b. En Nombre, escriba **`operand1`**.
    c. Active la opción Obligatorio.
    d. Mantenga Almacenamiento en caché desactivado.

    Repita los mismos pasos y cree una cadena de consulta llamada **`operand2`** y una cadena de consulta llamada **`operator`**.
15. Elija Crear método.

Ahora, vamos a crear una plantilla de mapeo para convertir las cadenas de consulta proporcionadas por el cliente en la carga de la solicitud de integración, tal y como requiere la función `Calc`. Esta plantilla mapea los tres parámetros de cadena de consulta declarados en Solicitud de método en

los valores de propiedad designados del objeto JSON como entrada a la función de Lambda del backend. El objeto JSON transformado se incluirá como la carga de la solicitud de integración.

Para mapear los parámetros de entrada a la solicitud de integración

1. En la pestaña Solicitud de integración, en Configuración de la solicitud de integración, seleccione Editar.
2. En Acceso directo de cuerpo de la solicitud, elija Cuando no haya plantillas definidas (recomendado).
3. Elija Plantillas de mapeo.
4. Elija Add mapping template (Añadir plantilla de asignación).
5. En Tipo de contenido, ingrese **`application/json`**.
6. En Cuerpo de la plantilla, introduzca el siguiente código:

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

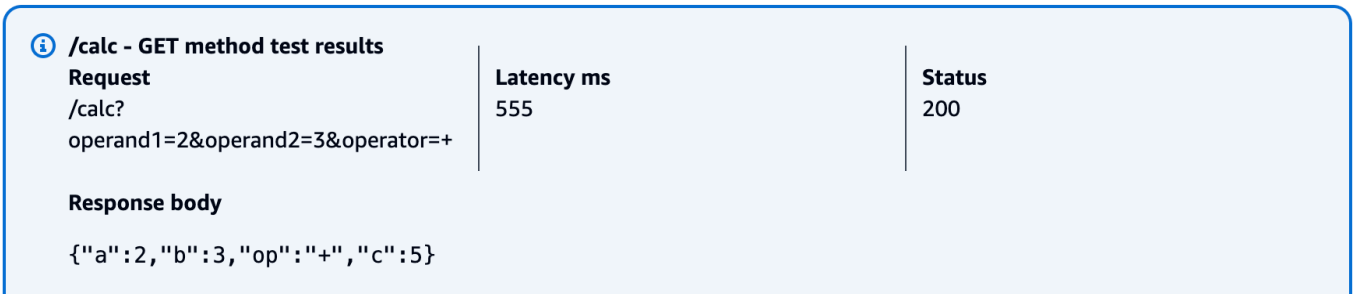
7. Seleccione Save.

Ahora puede probar el método GET para verificar que se ha configurado correctamente para invocar la función de Lambda.

Para probar el método **`GET`**

1. Elija la pestaña Prueba. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña.
2. En Cadenas de consulta, escriba **`operand1=2&operand2=3&operator=+`**.
3. Seleccione Test (Probar).

   El resultado debe ser parecido al siguiente:



Integración 2: Crear un método **POST** con una carga JSON para llamar a la función de Lambda

Al crear un método POST con una carga JSON para llamar a la función de Lambda, lo hace de tal manera que el cliente debe proporcionar la entrada necesaria a la función del backend en el cuerpo de la solicitud. Para garantizar que el cliente carga los datos de entrada correctos, habilitará la validación de solicitudes en la carga.

Para crear un método **POST** con una carga JSON

1. Seleccione el recurso calc y, a continuación, elija Crear método.
2. En Tipo de método, seleccione POST.
3. En Tipo de integración, seleccione Servicio de AWS.
4. En Región de AWS, seleccione la Región de AWS donde creó la función de Lambda.
5. En Servicio de AWS, seleccione Lambda.
6. Deje Subdominio de AWS en blanco.
7. En Método HTTP, seleccione POST.
8. En Tipo de acción, elija Usar sustitución de ruta. Esta opción nos permite especificar el ARN de la acción Invoke para ejecutar nuestra función `Calc`.
9. En Sustitución de ruta, escriba **`2015-03-31/functions/arn:aws:lambda:`*`us-east-2`*`:`*`account-id`*`:function:Calc/invocations`**. En *`account-id`*, introduzca su identificador de Cuenta de AWS. En *`us-east-2`*, introduzca la Región de AWS donde creó la función de Lambda.
10. En Rol de ejecución, escriba el ARN del rol para **`lambda_invoke_function_assume_apigw_role`**.
11. No cambie la configuración de Caché de credenciales ni Tiempo de espera predeterminado.
12. Elija Crear método.

Ahora vamos a crear un modelo de entrada para describir la estructura de los datos de entrada y validar el cuerpo de la solicitud entrante.

Para crear el modelo de entrada

1. En el panel de navegación principal, elija Modelos.
2. Seleccione Crear modelo.
3. En Nombre, escriba **`input`**.
4. En Tipo de contenido, ingrese **`application/json`**.

   Si no se encuentra ningún tipo de contenido coincidente, no se realiza la validación de la solicitud. Para utilizar el mismo modelo independientemente del tipo de contenido, introduzca **`$default`**.
5. En Esquema del modelo, escriba el siguiente modelo:

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

6. Seleccione Crear modelo.

Ahora cree un modelo de salida. Este modelo describe la estructura de datos de la salida calculada desde el backend. Se puede utilizar para asignar los datos de la respuesta de integración a un modelo diferente. Este tutorial se basa en el comportamiento de paso a través y no utiliza este modelo.

Para crear un modelo de salida

1. Seleccione Crear modelo.
2. En Nombre, escriba **`output`**.
3. En Tipo de contenido, ingrese **`application/json`**.

Si no se encuentra ningún tipo de contenido coincidente, no se realiza la validación de la solicitud. Para utilizar el mismo modelo independientemente del tipo de contenido, introduzca **`$default`**.

4. En Esquema del modelo, escriba el siguiente modelo:

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

5. Seleccione Crear modelo.

Ahora cree un modelo de resultados. Este modelo describe la estructura de datos de los datos de respuesta devueltos. Hace referencia a los esquemas de entrada y salida definidos en su API.

Para crear un modelo de resultados

1. Seleccione Crear modelo.
2. En Nombre, escriba **`result`**.
3. En Tipo de contenido, ingrese **`application/json`**.

   Si no se encuentra ningún tipo de contenido coincidente, no se realiza la validación de la solicitud. Para utilizar el mismo modelo independientemente del tipo de contenido, introduzca **`$default`**.
4. En Esquema del modelo, introduzca el siguiente modelo con su *`restapi-id`*. El *`restapi-id`* aparece entre paréntesis en la parte superior de la consola en el siguiente flujo: `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. Seleccione Crear modelo.

Ahora configure la solicitud de método de su método POST para habilitar la validación de la solicitud en el cuerpo de la solicitud entrante.

Habilitación de la validación de solicitudes en el método POST

1. En el panel de navegación principal, seleccione Recursos y, a continuación, seleccione el método POST en el árbol de recursos.
2. En la pestaña Solicitud de método, en Configuración de solicitud de método, elija Editar.
3. En Validador de solicitudes, seleccione Validar cuerpo.
4. Seleccione Cuerpo de la solicitud y, a continuación, seleccione Añadir modelo.
5. En Tipo de contenido, ingrese **`application/json`**.

   Si no se encuentra ningún tipo de contenido coincidente, no se realiza la validación de la solicitud. Para utilizar el mismo modelo independientemente del tipo de contenido, introduzca **`$default`**.
6. En Modelo, seleccione entrada.
7. Seleccione Save.

Ahora puede probar el método POST para verificar que se ha configurado correctamente para invocar la función de Lambda.

Para probar el método **`POST`**

1. Elija la pestaña Prueba. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña.
2. En Cuerpo de la solicitud, pegue la siguiente carga JSON.

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

3. Seleccione Probar

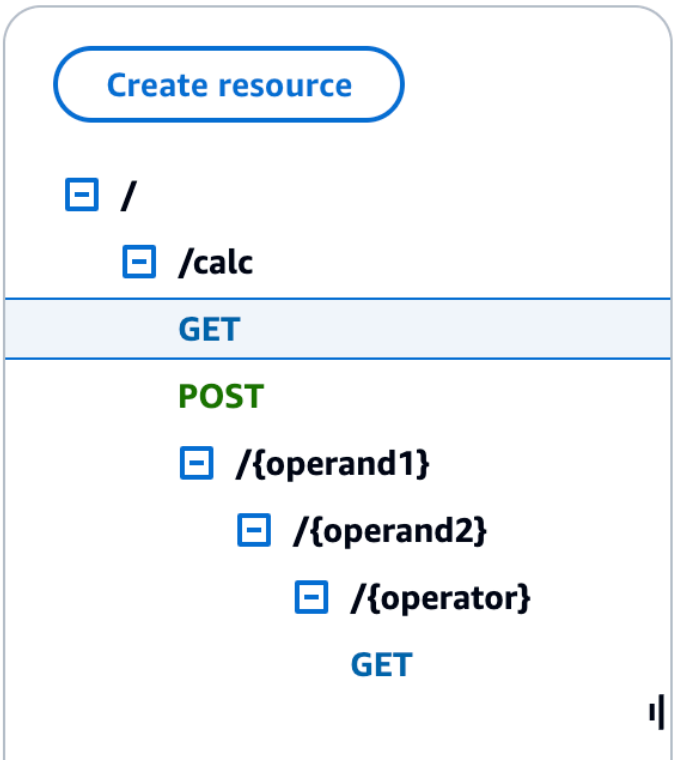
   Debería ver los siguientes datos de salida:

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

Integración 3: Crear un método **GET** con parámetros de ruta para llamar a la función de Lambda

Ahora creará un método GET en un recurso especificado por una secuencia de parámetros de ruta para llamar a la función de Lambda del backend. Los valores de los parámetros de ruta especifican los datos de entrada a la función de Lambda. Definirá una plantilla de asignación para asignar los valores de los parámetros de ruta entrantes a la carga de la solicitud de integración necesaria.

La estructura de recursos de la API resultante será como la siguiente:



Para crear un recurso /{operand1}/{operand2}/{operator}

1. Elija Crear recurso.
2. En Ruta de recurso, seleccione `/calc`.
3. En Nombre del recurso, escriba **`{operand1}`**.
4. Mantenga desactivado CORS (uso compartido de recursos entre orígenes).
5. Elija Crear recurso.
6. En Ruta de recurso, seleccione `/calc/{operand1}/`.
7. En Nombre del recurso, escriba **`{operand2}`**.
8. Mantenga desactivado CORS (uso compartido de recursos entre orígenes).
9. Elija Crear recurso.
10. En Ruta de recurso, seleccione `/calc/{operand1}/{operand2}/`.
11. En Nombre del recurso, escriba **`{operator}`**.
12. Mantenga desactivado CORS (uso compartido de recursos entre orígenes).
13. Elija Crear recurso.

Esta vez utilizará la integración de Lambda incorporada en la consola de API Gateway para configurar la integración del método.

Para configurar una integración del método

1. Seleccione el recurso /{operand1}/{operand2}/{operator} y, a continuación, elija Crear método.
2. En Tipo de método, seleccione GET.
3. En Tipo de integración, seleccione Lambda.
4. Mantenga desactivada la Integración de proxy Lambda.
5. En Función de Lambda, seleccione la Región de AWS donde creó su función de Lambda e introduzca **`Calc`**.
6. Mantenga activado Tiempo de espera predeterminado.
7. Elija Crear método.

Ahora debe crear una plantilla de mapeo para mapear los tres parámetros de la ruta de la URL, declarados cuando se creó el recurso /calc/{operand1}/{operand2}/{operator}, con los valores de propiedad designados en el objeto JSON. Como las rutas URL deben estar codificadas como direcciones URL, el operador de división debe especificarse como `%2F` en lugar de `/`. Esta plantilla traduce `%2F` en `'/'` antes de pasarlo a la función de Lambda.

Para crear una plantilla de mapeo

1. En la pestaña Solicitud de integración, en Configuración de la solicitud de integración, seleccione Editar.
2. En Acceso directo de cuerpo de la solicitud, elija Cuando no haya plantillas definidas (recomendado).
3. Elija Plantillas de mapeo.
4. En Tipo de contenido, ingrese **`application/json`**.
5. En Cuerpo de la plantilla, introduzca el siguiente código:

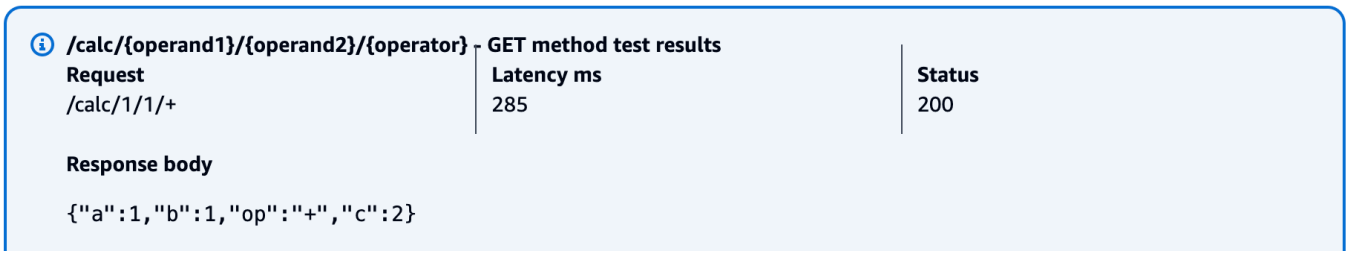
```
{
   "a": "$input.params('operand1')",
   "b": "$input.params('operand2')",
   "op":
 #if($input.params('operator')=='%2F')"/"#{else}"$input.params('operator')"#end
}
```

6. Seleccione Save.

Ahora puede probar el método `GET` para verificar que se ha configurado correctamente para invocar la función de Lambda y pasar el resultado original a través de la respuesta de integración sin mapear.

Para probar el método **`GET`**

1. Elija la pestaña Prueba. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña.
2. Para Ruta, haga lo siguiente:
   a. En operand1, introduzca **1**.
   b. En operand2, introduzca **1**.
   c. En operator, introduzca **+**.
3. Seleccione Test (Probar).
4. El resultado debe tener el siguiente aspecto:



A continuación, creará el modelo de la estructura de datos de la carga de la respuesta del método detrás del esquema `result`.

De forma predeterminada, el cuerpo de la respuesta del método se asigna a un modelo vacío. Esto hará que el cuerpo de la respuesta de integración se transfiera sin asignación. Sin embargo, cuando genere un SDK para alguno de los lenguajes con establecimiento inflexible de tipos, como Java u Objective-C, los usuarios del SDK recibirán un objeto vacío como resultado. Para garantizar que el cliente REST y los clientes del SDK reciban el resultado deseado, debe crear los datos de la respuesta mediante un esquema predefinido. A continuación, definirá un modelo para el cuerpo de la respuesta del método y cómo crear una plantilla de asignación para traducir el cuerpo de la respuesta de integración al cuerpo de la respuesta del método.

Para crear una respuesta del método

1. En la pestaña Respuesta del método, en Respuesta 200, elija Editar.
2. En Cuerpo de la respuesta, seleccione Agregar modelo.
3. En Tipo de contenido, ingrese **`application/json`**.
4. En Modelo, seleccione resultado.
5. Seleccione Save.

Al configurar el modelo para el cuerpo de la respuesta del método, nos aseguramos de que los datos de la respuesta se emitan en el objeto `result` de un determinado SDK. Para asegurarse de que los datos de la respuesta de integración se asignen según corresponda, necesitará una plantilla de asignación.

Para crear una plantilla de mapeo

1. En la pestaña Respuesta de integración, en Predeterminado: respuesta, seleccione Editar.
2. Elija Plantillas de mapeo.
3. En Tipo de contenido, ingrese **`application/json`**.
4. En Cuerpo de la plantilla, introduzca el siguiente código:

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

5. Seleccione Save.

Para probar la plantilla de mapeo

1. Elija la pestaña Prueba. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña.

2. Para Ruta, haga lo siguiente:

   a. En operand1, introduzca **1**.

   b. En operand2, introduzca **2**.

   c. En operator, introduzca **+**.

3. Seleccione Probar

4. El resultado será parecido al siguiente:

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

En este momento, solo puede llamar a la API utilizando la característica Prueba de la consola de API Gateway. Para hacer que esté disponible para los clientes, debe implementar la API. Asegúrese siempre para volver a implementar la API, cada vez que agrega, modifica o elimina un recurso o un método, actualizar una asignación de datos o actualiza la configuración de la etapa. De lo contrario, las nuevas características o actualizaciones no estarán disponibles para los clientes de la API.

Para implementar la API

1. Elija Implementar API.

2. En Etapa, seleccione Nueva etapa.

3. En Stage name (Nombre de etapa), escriba **`Prod`**.

4. (Opcional) En Description (Descripción), introduzca una descripción.

5. Elija Implementar.

6. (Opcional) En Detalles de la etapa, para URL de invocación, puede elegir el icono de copia para copiar la URL de invocación de su API. Puede utilizar esto con herramientas como Postman y cURL para probar la API.

> **Note**
>
> Vuelva a implementar la API cada vez que agrega, modifica o elimina un recurso o un método, actualiza una asignación de datos o actualiza la configuración de la etapa. De lo contrario, las nuevas características o actualizaciones no estarán disponibles para los clientes de la API.

Definiciones de OpenAPI de una API de ejemplo integrada con una función de Lambda

OpenAPI 2.0

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

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

```
                "application/json": "#set($res = $input.path('$'))\n{\n    \"result
\": \"$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: Creación de una API de REST como proxy de Amazon S3

Para ilustrar cómo usar una API de REST en API Gateway como proxy de Amazon S3, en esta sección se describe cómo crear y configurar una API de REST para exponer las siguientes operaciones de Amazon S3:

- Exponer GET en el recurso raíz de la API para enumerar todos los buckets de Amazon S3 de un intermediario.
- Exponer GET en un recurso de carpeta para ver una lista de todos los objetos de un bucket de Amazon S3.
- Exponer GET en un recurso de carpeta o elemento para ver o descargar un objeto de un bucket de Amazon S3.

Es posible que desee importar la API de ejemplo como proxy de Amazon S3, tal y como se muestra en Definiciones de OpenAPI de la API de ejemplo como un proxy de Amazon S3. Este ejemplo contiene métodos más expuestos. Para obtener instrucciones sobre cómo importar una API mediante la definición de OpenAPI, consulte Desarrollo de API de REST mediante OpenAPI en API Gateway.

> **Note**
>
> Para integrar su API de API Gateway con Amazon S3, debe elegir una región en la que estén disponibles los servicios API Gateway y Amazon S3. Para obtener información sobre la disponibilidad en función de la región, consulte Cuotas y puntos de conexión de Amazon API Gateway.

Temas

- Configurar permisos de IAM para que la API invoque acciones de Amazon S3
- Crear recursos de API para representar recursos de Amazon S3
- Exponer un método de API para enumerar los buckets de Amazon S3 del intermediario
- Exponer métodos de API para tener acceso a un bucket de Amazon S3
- Exponer métodos de API para tener acceso a un objeto de Amazon S3 en un bucket
- Definiciones de OpenAPI de la API de ejemplo como un proxy de Amazon S3
- Llamada a la API mediante un cliente API de REST

Configurar permisos de IAM para que la API invoque acciones de Amazon S3

Para permitir que la API invoque las acciones de Amazon S3, debe tener las políticas de IAM adecuadas asociadas a un rol de IAM. En este paso, deberá crear un nuevo rol de IAM.

Para crear el rol de ejecución del proxy de servicio de AWS

1. Inicie sesión en Consola de administración de AWS y abra la consola IAM en https://console.aws.amazon.com/iam/.
2. Elija Roles.
3. Elija Creación de rol.
4. Elija Servicio de AWS en Seleccionar el tipo de entidad de confianza y, a continuación, elija API Gateway y seleccione Permite que API Gateway envíe registros a CloudWatch Logs.
5. Seleccione Siguiente y de nuevo Siguiente.
6. En Nombre de rol, escriba **`APIGatewayS3ProxyPolicy`** y luego elija Crear rol.
7. En la lista Roles, elija el rol que acaba de crear. Puede que tenga que desplazarse o usar la barra de búsqueda para encontrar el rol.
8. Para el rol seleccionado, seleccione la pestaña Agregar permisos.
9. Elija Adjuntar políticas en la lista desplegable.
10. En la barra de búsqueda, escriba **`AmazonS3FullAccess`** y, a continuación, elija Añadir permisos.

> **Note**
>
> Para simplificar el proceso, este tutorial utiliza una política administrada. Como práctica recomendada, se deben crear políticas de IAM propias para otorgar los permisos mínimos requeridos.

11. Anote el ARN del rol recién creado, ya que lo usará más adelante.

Crear recursos de API para representar recursos de Amazon S3

Debe utilizar el recurso raíz (`/`) de la API como contenedor de buckets de Amazon S3 de un intermediario autenticado. También debe crear los recursos `Folder` y `Item` para representar un bucket de Amazon S3 determinado y un objeto de Amazon S3 determinado, respectivamente. El intermediario especificará el nombre de carpeta y la clave de objeto en forma de parámetros de ruta como parte de la URL de una solicitud.

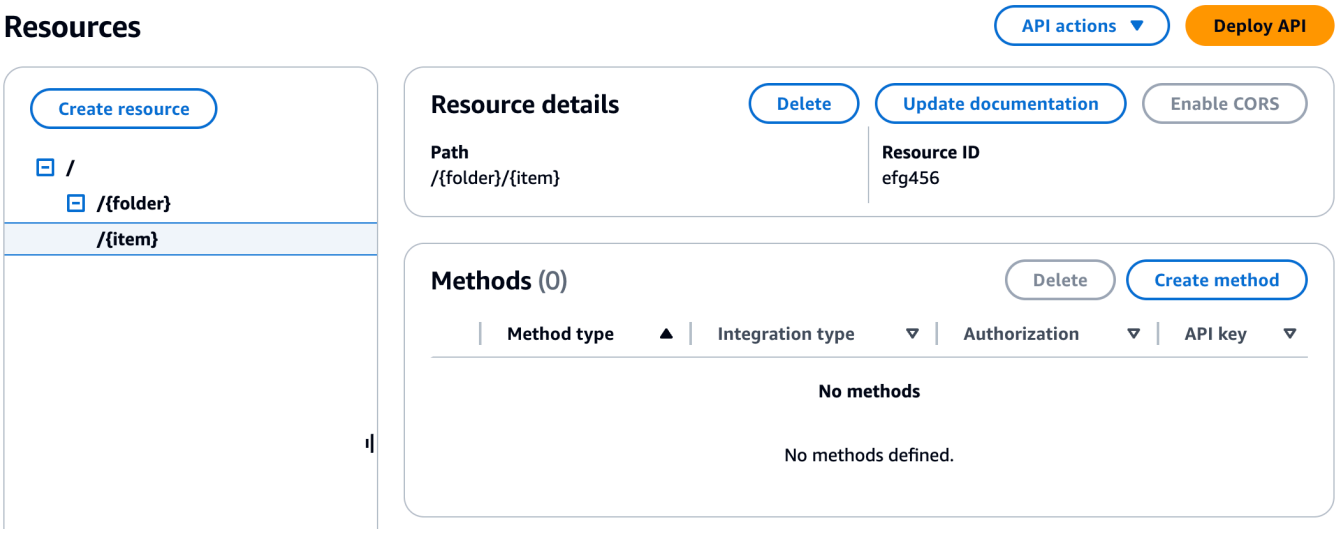
> **Note**
>
> Al acceder a los objetos cuya clave de objeto incluye `/` o cualquier otro carácter especial, el carácter estar codificado en la URL. Por ejemplo, `test/test.txt` deben codificarse en `test%2Ftest.txt`.

Para crear un recurso de API que exponga las características del servicio Amazon S3

1. En la misma Región de AWS que creó el bucket de Amazon S3, cree una API llamada MyS3. Este recurso raíz de la API (/) representa el servicio Amazon S3. En este paso, creará dos recursos adicionales,/{folder} y /{item}.
2. Elija Crear recurso.
3. Mantenga Recurso proxy desactivado.
4. En Ruta de recurso, seleccione `/`.
5. En Nombre del recurso, escriba **`{folder}`**.
6. Mantenga desactivado CORS (uso compartido de recursos entre orígenes).
7. Elija Crear recurso.
8. Seleccione el recurso /{folder} y, a continuación, elija Crear recurso.

9. Siga los pasos anteriores para crear un recurso secundario de /{folder} denominado **`{item}`**.

   Su API final debería parecerse a la siguiente:



Exponer un método de API para enumerar los buckets de Amazon S3 del intermediario

Obtener la lista de buckets de Amazon S3 del intermediario implica invocar la acción GET Service en Amazon S3. En el recurso raíz de la API, (/), cree el método GET. Configure el método GET para realizar la integración con Amazon S3, tal y como se indica a continuación.

Para crear e inicializar el método **`GET /`** de la API

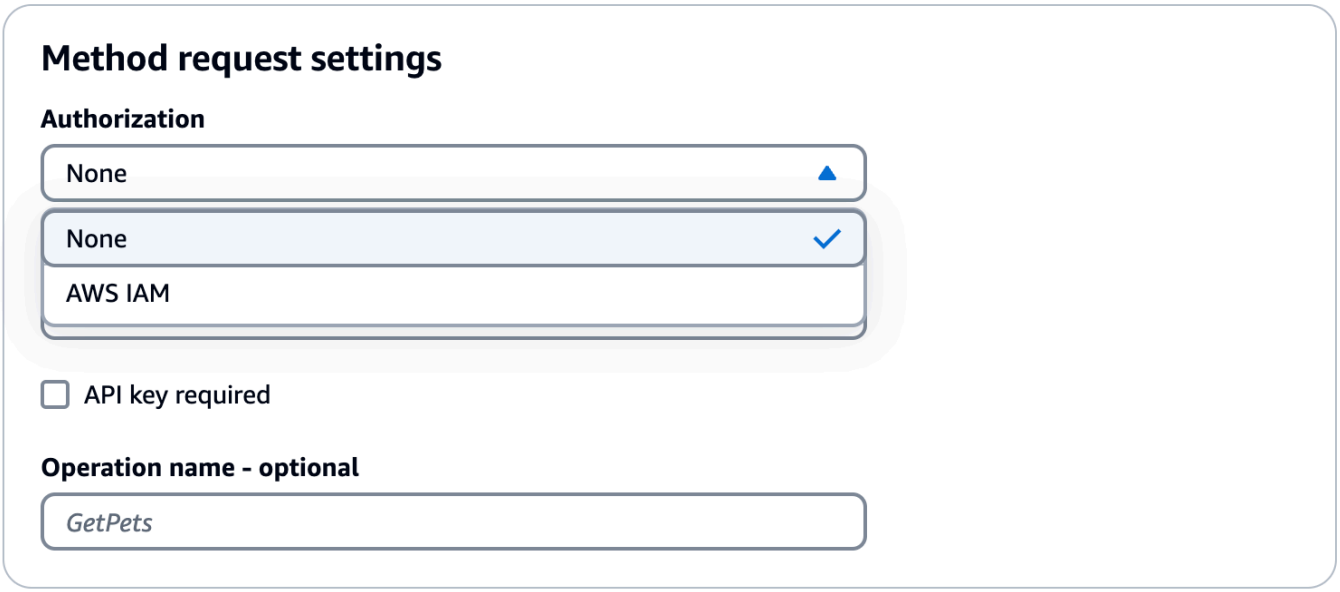
1. Seleccione el recurso / y, a continuación, elija Crear método.
2. En el tipo de método, seleccione GET.
3. En Tipo de integración, seleccione Servicio de AWS.
4. Para Región de AWS, seleccione la Región de AWS donde creó su bucket de Amazon S3.
5. Para Servicio de AWS, elija Amazon Simple Storage Service.
6. Deje Subdominio de AWS en blanco.
7. En Método HTTP, seleccione GET.
8. En Tipo de acción, elija Usar sustitución de ruta.

   Al anular la ruta, API Gateway reenvía la solicitud del cliente a Amazon S3 como la solicitud del tipo de ruta de la API de REST de Amazon S3 correspondiente, en la cual un recurso de

Amazon S3 se expresa por la ruta de recursos del patrón `s3-host-name/bucket/key`. API Gateway establece el `s3-host-name` y transmite el `bucket` y la `key` especificados por el cliente a Amazon S3.

9. En Sustitución de ruta, escriba /.
10. En Rol de ejecución, escriba el ARN del rol para **`APIGatewayS3ProxyPolicy`**.
11. Elija Configuración de solicitud de método.

    Debe utilizar la configuración de solicitud de método para controlar quién puede llamar a este método de la API.
12. En Autorización, en el menú desplegable, seleccione `AWS_IAM`.



13. Elija Crear método.

Esta configuración integra la solicitud `GET https://`*`your-api-host`*`/`*`stage`*`/` del frontend con la solicitud `GET https://`*`your-s3-host`*`/` del backend.

Para que la API devuelva respuestas correctas y excepciones al intermediario, debe declarar las respuestas 200, 400 y 500 en Respuesta de método. Debe utilizar la asignación predeterminada para las respuestas 200, de forma que las respuestas del backend del código de estado no declarado aquí se devuelvan al intermediario como respuestas 200.

Para declarar tipos de respuestas para el método **`GET /`**

1. En la pestaña Respuesta del método, en Respuesta 200, elija Editar.

2. Elija Añadir encabezado y haga lo siguiente:

   a. En Nombre de encabezado, escriba **`Content-Type`**.

   b. Elija Agregar encabezado.

   Repita estos pasos para crear un encabezado **`Timestamp`** y un encabezado **`Content-Length`**.

3. Seleccione Save.
4. En la pestaña Método de respuesta, en Respuestas de método, seleccione Crear respuesta.
5. En Código de estado HTTP, introduzca 400.

   No establezca ningún encabezado para esta respuesta.

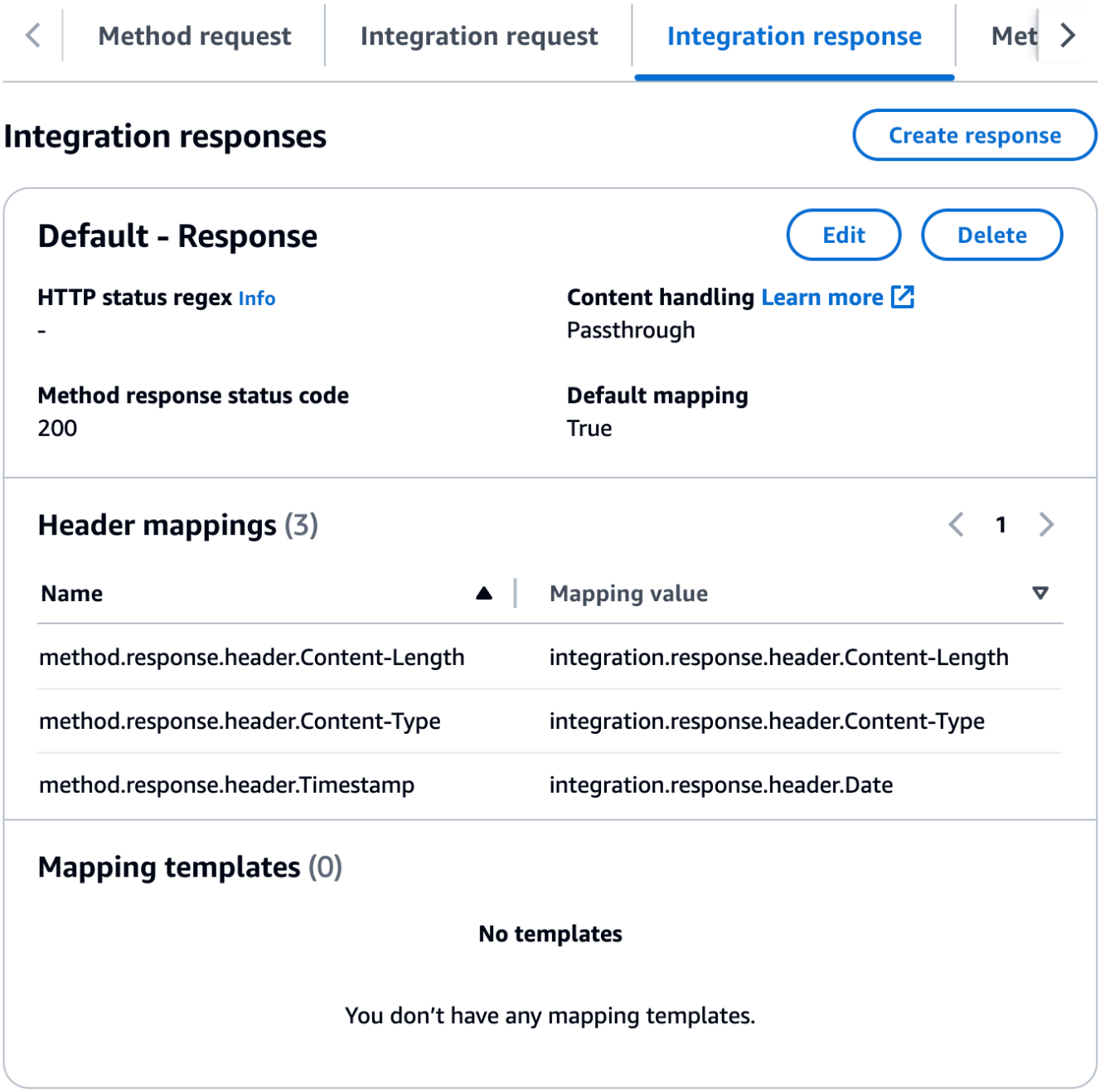
6. Seleccione Save.
7. Repita los pasos siguientes para crear la respuesta 500.

   No establezca ningún encabezado para esta respuesta.

Como la respuesta de integración correcta de Amazon S3 devuelve la lista de buckets como una carga XML y la respuesta de método predeterminada de API Gateway devuelve una carga JSON, debe mapear el valor del parámetro del encabezado Content-Type del backend con su homólogo en el frontend. De lo contrario, el cliente recibirá `application/json` para el tipo de contenido cuando el cuerpo de la respuesta sea en realidad una cadena XML. El siguiente procedimiento muestra cómo realizar esta configuración. Además, también quiere mostrar al cliente otros parámetros de encabezado, como Date y Content-Length.

Para configurar asignaciones de encabezado de respuesta para el método GET /

1. En la pestaña Respuesta de integración, en Predeterminado: respuesta, seleccione Editar.
2. Para el encabezado Content-Length, introduzca **`integration.response.header.Content-Length`** en el valor de mapeo.
3. Para el encabezado Content-Type, introduzca **`integration.response.header.Content-Type`** en el valor de mapeo.
4. Para el encabezado Timestamp, introduzca **`integration.response.header.Date`** en el valor de mapeo.
5. Seleccione Save. El resultado debería ser similar al siguiente:



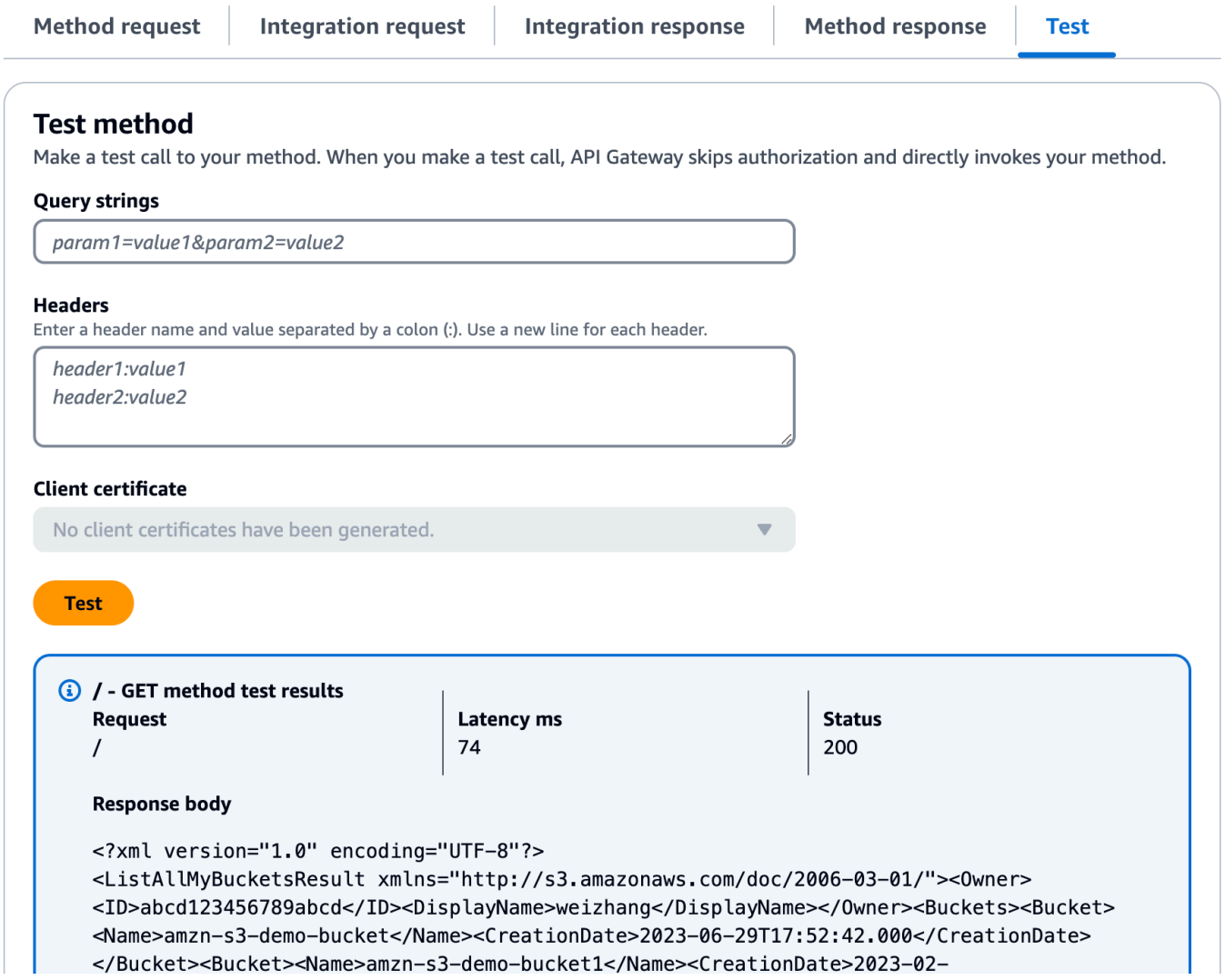
6. En la pestaña Respuesta de integración, en Respuestas de integración, seleccione Crear respuesta.
7. En Expresión regular de estado de HTTP, escriba **`4\d{2}`**. Esto asigna todos los códigos de estado de respuesta HTTP 4xx a la respuesta del método.
8. En Código de estado de respuesta del método, seleccione **`400`**.
9. Seleccione Crear.

10. Repita los siguientes pasos para crear una respuesta de integración para la respuesta del método 500. En Expresión regular de estado de HTTP, escriba **`5\d{2}`**.

Como práctica recomendada, puede probar la API tal y como está configurada de momento.

Para probar el método **`GET /`**

1. Elija la pestaña Prueba. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña.
2. Seleccione Probar El resultado debe ser similar al de la siguiente imagen:



Exponer métodos de API para tener acceso a un bucket de Amazon S3

Para trabajar con un bucket de Amazon S3, exponga el método `GET` en el recurso /{folder} para mostrar los objetos de un bucket. Las instrucciones son similares a las que se describen en Exponer un método de API para enumerar los buckets de Amazon S3 del intermediario. Para ver más métodos, puede importar la API de ejemplo aquí, Definiciones de OpenAPI de la API de ejemplo como un proxy de Amazon S3.

Para exponer el método GET en un recurso de carpeta

1. Seleccione el recurso /{folder} y, a continuación, elija Crear método.
2. En el tipo de método, seleccione GET.
3. En Tipo de integración, seleccione Servicio de AWS.
4. Para Región de AWS, seleccione la Región de AWS donde creó su bucket de Amazon S3.
5. Para Servicio de AWS, elija Amazon Simple Storage Service.
6. Deje Subdominio de AWS en blanco.
7. En Método HTTP, seleccione GET.
8. En Tipo de acción, elija Usar sustitución de ruta.
9. En Sustitución de ruta, escriba **`{bucket}`**.
10. En Rol de ejecución, escriba el ARN del rol para **`APIGatewayS3ProxyPolicy`**.
11. Elija Crear método.

Establezca el parámetro de ruta `{folder}` en la URL del punto de conexión de Amazon S3. Debe mapear el parámetro de ruta `{folder}` de la solicitud de método al parámetro de ruta `{bucket}` de la solicitud de integración.

Para mapear **`{folder}`** a **`{bucket}`**:

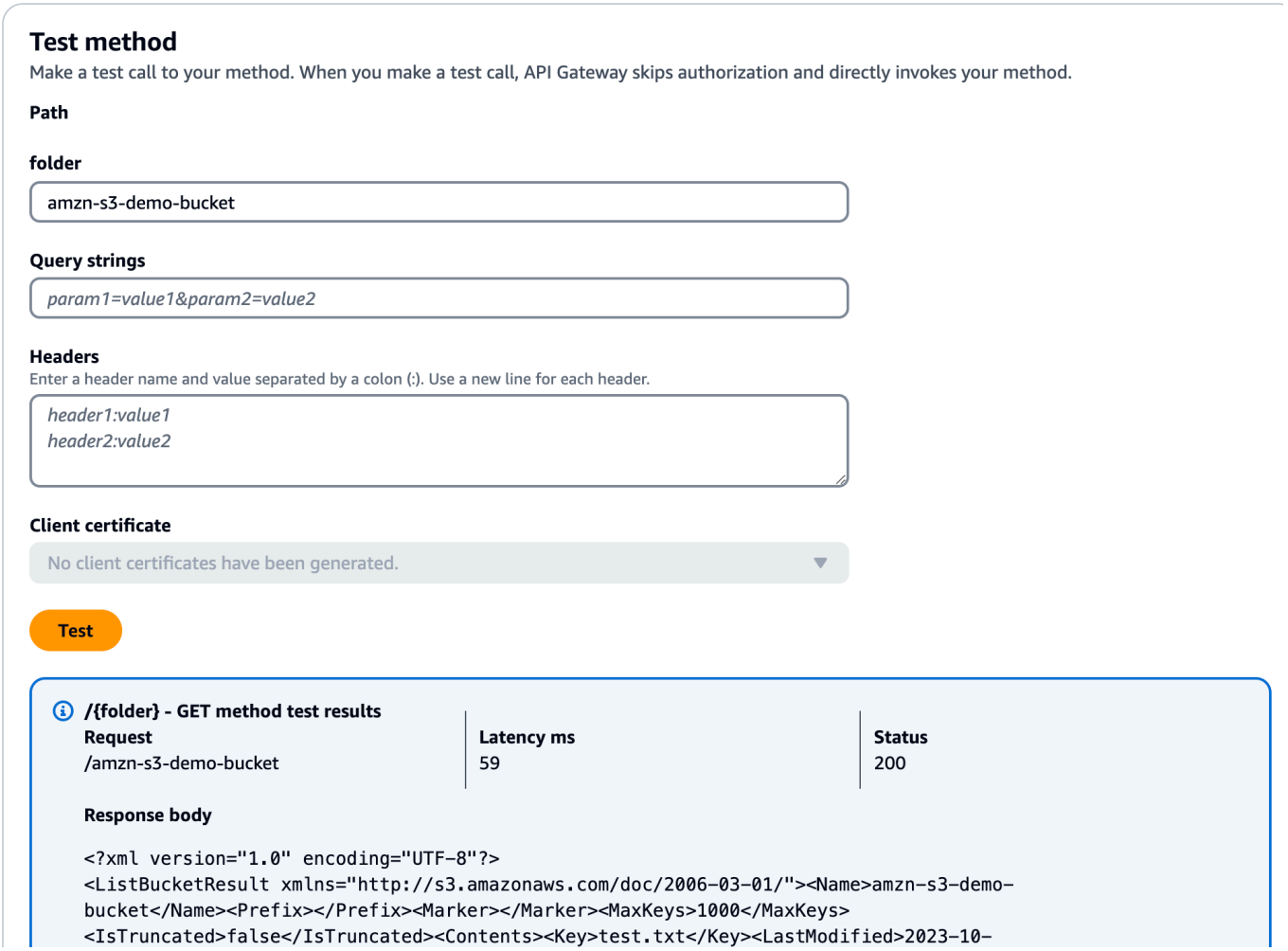
1. En la pestaña Solicitud de integración, en Configuración de la solicitud de integración, seleccione Editar.
2. Elija Parámetros de la ruta URL y, a continuación, elija Agregar parámetro de ruta.
3. En Nombre, escriba **`bucket`**.
4. En Asignado desde, introduzca **`method.request.path.folder`**.
5. Seleccione Save.

Ahora, pruebe la API.

Para probar el método **`/{folder} GET`**

1. Elija la pestaña Prueba. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña.
2. En Ruta, para la carpeta, introduzca el nombre de tu bucket.
3. Seleccione Probar

   El resultado de la prueba incluirá una lista de los objetos del bucket.



Exponer métodos de API para tener acceso a un objeto de Amazon S3 en un bucket

Amazon S3 admite las acciones GET, DELETE, HEAD, OPTIONS, POST y PUT para acceder a objetos de un bucket determinado y administrarlos. En este tutorial, expondrá un método GET del

recurso `{folder}/{item}` para obtener una imagen de un bucket. Para ver más aplicaciones del recurso `{folder}/{item}`, consulte la API de ejemplo, Definiciones de OpenAPI de la API de ejemplo como un proxy de Amazon S3.

Para exponer el método GET en un recurso de elemento

1. Seleccione el recurso /{item} y, a continuación, elija Crear método.
2. En el tipo de método, seleccione GET.
3. En Tipo de integración, seleccione Servicio de AWS.
4. Para Región de AWS, seleccione la Región de AWS donde creó su bucket de Amazon S3.
5. Para Servicio de AWS, elija Amazon Simple Storage Service.
6. Deje Subdominio de AWS en blanco.
7. En Método HTTP, seleccione GET.
8. En Tipo de acción, elija Usar sustitución de ruta.
9. En Sustitución de ruta, introduzca {bucket}/{object}.
10. En Rol de ejecución, escriba el ARN del rol para **`APIGatewayS3ProxyPolicy`**.
11. Elija Crear método.

Establezca los parámetros de ruta `{folder}` y `{item}` en la URL del punto de conexión de Amazon S3. Debe mapear el parámetro de ruta de la solicitud de método al parámetro de ruta de la solicitud de integración.

En este paso, hará lo siguiente:

- Mapee el parámetro de ruta `{folder}` de la solicitud de método al parámetro de ruta `{bucket}` de la solicitud de integración.
- Mapee el parámetro de ruta `{item}` de la solicitud de método al parámetro de ruta `{object}` de la solicitud de integración.

Para mapear **`{folder}`** a **`{bucket}`** y **`{item}`** a **`{object}`**

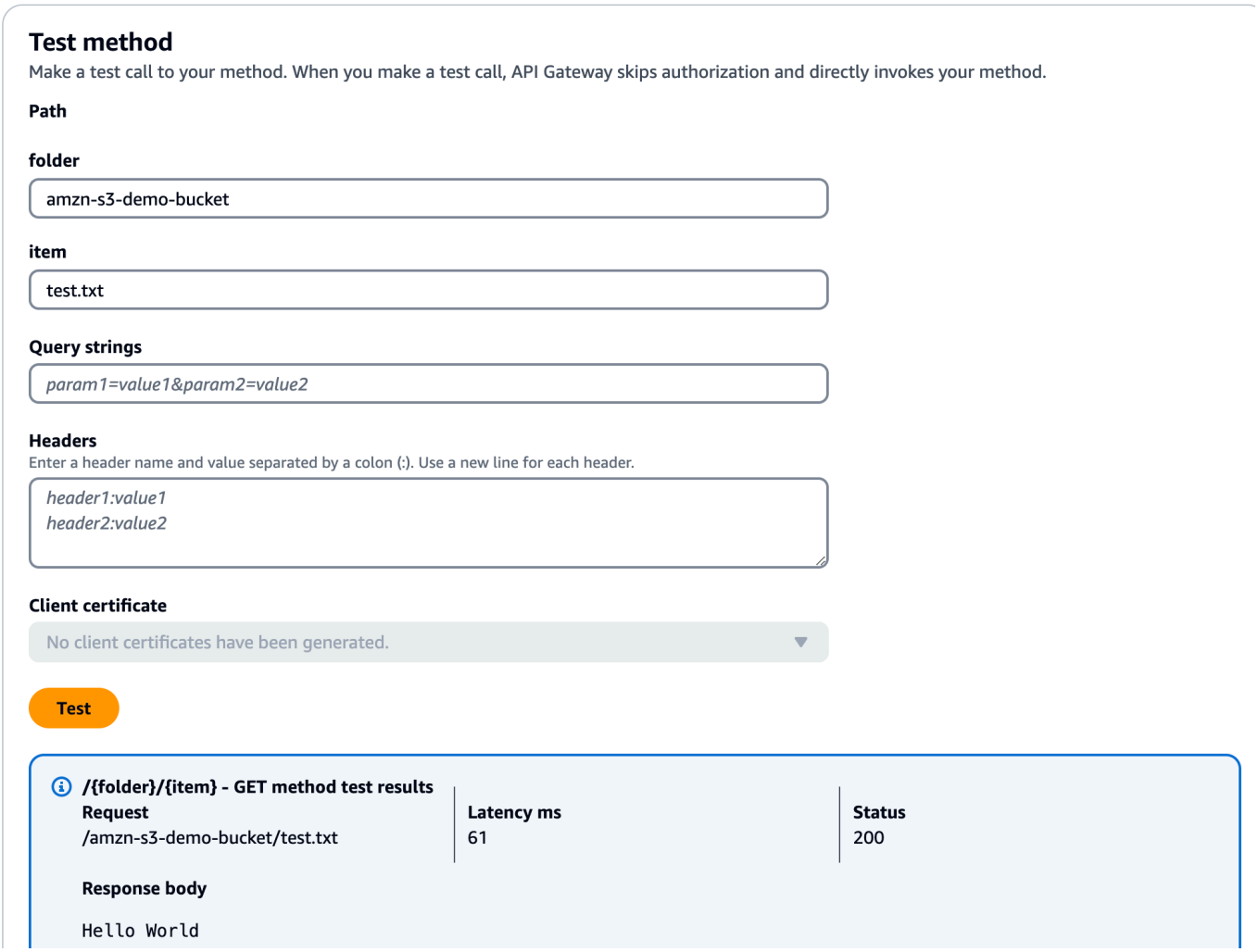
1. En la pestaña Solicitud de integración, en Configuración de la solicitud de integración, seleccione Editar.
2. Elija los Parámetros de la ruta URL.
3. Elija Añadir parámetro de ruta.

4. En Nombre, escriba **`bucket`**.
5. En Asignado desde, introduzca **`method.request.path.folder`**.
6. Elija Añadir parámetro de ruta.
7. En Nombre, escriba **`object`**.
8. En Asignado desde, introduzca **`method.request.path.item`**.
9. Seleccione Save.

Para probar el método **`/{folder}/{item} GET`**

1. Elija la pestaña Prueba. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña.
2. En Ruta, para la carpeta, introduzca el nombre de tu bucket.
3. En Ruta, para el elemento, introduzca el nombre de un elemento.
4. Seleccione Probar

   El cuerpo de la respuesta incluirá el contenido del elemento.



La solicitud devuelve correctamente el texto sin formato ("Hello world") como el contenido del archivo especificado (test.txt) en el bucket de Amazon S3 indicado (amzn-s3-demo-bucket).

Para descargar o cargar archivos binarios, que en API Gateway se considera cualquier cosa que no sea contenido JSON codificado en UTF-8, es necesaria una configuración adicional de la API. Esto se detalla de la siguiente manera:

Descargar o cargar archivos binarios de S3

1. Registrar los tipos de archivo del archivo afectado en los tipos "binaryMediaTypes" de la API. Puede hacer lo siguiente en la consola:

   a. Elija Configuración de API para la API.

   b. En Tipos de medios binarios, seleccione Gestionar tipos de medios.

c. Seleccione Añadir tipo de medio binario y, a continuación, introduzca el tipo de medio requerido, por ejemplo, `image/png`.

d. Elija Guardar cambios, para guardar la configuración.

2. Añada las cabeceras `Content-Type` (para cargar) y/o `Accept` (para descargar) a la solicitud del método para exigir que el cliente especifique el tipo de medios binarios necesarios y asignarlos a la solicitud de integración.

3. Establezca la opción Content Handling (Tratamiento de contenido) en `Passthrough` en la solicitud de integración (para cargar) y en una respuesta de integración (para descargar). Asegúrese de que no se define ninguna plantilla de mapeo para el tipo de contenido afectado. Para obtener más información, consulte the section called “Transformaciones de datos”.

El límite de tamaño de carga es 10 MB. Consulte Cuotas para configurar y ejecutar una API de REST en API Gateway.

Asegúrese de que los archivos en Amazon S3 incluyen los tipos de contenido correctos en los metadatos. Para contenido multimedia que se puede transmitir, es posible que los metadatos deban incluir `Content-Disposition:inline`.

Para obtener más información sobre la compatibilidad con datos binarios en API Gateway, consulte Conversiones de tipo de contenido en API Gateway.

Definiciones de OpenAPI de la API de ejemplo como un proxy de Amazon S3

En las siguientes definiciones de OpenAPI, se describe una API que funciona como proxy de Amazon S3. Esta API contiene más operaciones de Amazon S3 que la API que creó en el tutorial. Los siguientes métodos se exponen en las definiciones de OpenAPI:

- Exponer GET en el recurso raíz de la API para enumerar todos los buckets de Amazon S3 de un intermediario.
- Exponer GET en un recurso de carpeta para ver una lista de todos los objetos de un bucket de Amazon S3.
- Exponer PUT en un recurso de carpeta para agregar un bucket a Amazon S3.
- Exponer DELETE en un recurso de carpeta para eliminar un bucket de Amazon S3.
- Exponer GET en un recurso de carpeta o elemento para ver o descargar un objeto de un bucket de Amazon S3.

- Exponer PUT en un recurso de carpeta o elemento para cargar un objeto en un bucket de Amazon S3.
- Exponer HEAD en un recurso de carpeta o elemento para obtener los metadatos de objeto en un bucket de Amazon S3.
- Exponer DELETE en un recurso de carpeta o elemento para eliminar un objeto de un bucket de Amazon S3.

Para obtener instrucciones sobre cómo importar una API mediante la definición de OpenAPI, consulte Desarrollo de API de REST mediante OpenAPI en API Gateway.

Para obtener instrucciones sobre cómo crear una API similar, consulte Tutorial: Creación de una API de REST como proxy de Amazon S3.

Para obtener información sobre cómo invocar esta API mediante Postman, que admite la autorización de IAM AWS, consulte Llamada a la API mediante un cliente API de REST.

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"
      }
    }
  }
}
```

Llamada a la API mediante un cliente API de REST

Para proporcionar un tutorial completo, ahora mostraremos cómo llamar a la API mediante Postman, que admite la autorización de IAM de AWS.

Para llamar a nuestra API de proxy de Amazon S3 mediante Postman

1. Implemente o vuelva a implementar la API. Anote la URL base de la API que se muestra junto a Invoke URL (URL de invocación) en la parte superior de Stage Editor (Editor de etapas).
2. Inicie Postman.
3. Elija Autorización y, a continuación, elija `AWS Signature`. Ingrese el ID de clave de acceso y la clave de acceso secreta del usuario de IAM en los campos de entrada AccessKey y SecretKey,

respectivamente. Ingrese la Región de AWS en la que va a implementar la API en el cuadro de texto Región de AWS. Ingrese `execute-api` en el campo de entrada Nombre del servicio.

Puede crear un par de claves desde la pestaña Credenciales de seguridad de su cuenta de usuario de IAM en la consola de administración de IAM.

4. Para agregar un bucket denominado `amzn-s3-demo-bucket` a su cuenta de Amazon S3 en la región *`{region}`*:

   a. Seleccione PUT en la lista desplegable de métodos y escriba la URL del método (`https://`*`api-id`*`.execute-api.`*`aws-region`*`.amazonaws.com/`*`stage`*`/`*`folder-name`*)

   b. Establezca el valor del encabezado `Content-Type` en `application/xml`. Es posible que necesite eliminar todos los encabezados existentes antes de configurar el tipo de contenido.

   c. Elija el elemento de menú Body (Cuerpo) y escriba el siguiente fragmento XML como cuerpo de la solicitud:

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

   d. Elija Send (Enviar) para enviar la solicitud. Si todo va bien, debería recibir una respuesta `200 OK` con una carga vacía.

5. Para añadir un archivo de texto a un bucket, siga las instrucciones anteriores. Si especifica el nombre de bucket de **`amzn-s3-demo-bucket`** para `{folder}` y el nombre de archivo **`Readme.txt`** para `{item}` en la dirección URL y proporciona la cadena de texto **`Hello, World!`** como contenido de archivo (con lo que se convierte en la carga de la solicitud), la respuesta será

   ```
   PUT /S3/amzn-s3-demo-bucket/Readme.txt HTTP/1.1
   Host: 9gn28ca086.execute-api.{region}.amazonaws.com
   Content-Type: application/xml
   X-Amz-Date: 20161015T062647Z
   Authorization: AWS4-HMAC-SHA256 Credential=access-key-id/20161015/{region}/execute-
   api/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
    Signature=ccadb877bdb0d395ca38cc47e18a0d76bb5eaf17007d11e40bf6fb63d28c705b
   Cache-Control: no-cache
   Postman-Token: 6135d315-9cc4-8af8-1757-90871d00847e
   ```

```
Hello, World!
```

Si todo sale bien, debe recibir una respuesta `200 OK` con una carga vacía.

6. Para obtener el contenido del archivo `Readme.txt` que acaba de añadir al bucket `amzn-s3-demo-bucket`, envíe una solicitud GET como la siguiente:

```
GET /S3/amzn-s3-demo-bucket/Readme.txt HTTP/1.1
Host: 9gn28ca086.execute-api.{region}.amazonaws.com
Content-Type: application/xml
X-Amz-Date: 20161015T063759Z
Authorization: AWS4-HMAC-SHA256 Credential=access-key-id/20161015/{region}/
execute-api/aws4_request, SignedHeaders=content-type;host;x-amz-date,
 Signature=ba09b72b585acf0e578e6ad02555c00e24b420b59025bc7bb8d3f7aed1471339
Cache-Control: no-cache
Postman-Token: d60fcb59-d335-52f7-0025-5bd96928098a
```

Si todo sale bien, debe recibir una respuesta `200 OK` con la cadena de texto `Hello, World!` como la carga.

7. Para mostrar los elementos del bucket `amzn-s3-demo-bucket`, envíe la siguiente solicitud:

```
GET /S3/amzn-s3-demo-bucket HTTP/1.1
Host: 9gn28ca086.execute-api.{region}.amazonaws.com
Content-Type: application/xml
X-Amz-Date: 20161015T064324Z
Authorization: AWS4-HMAC-SHA256 Credential=access-key-id/20161015/{region}/
execute-api/aws4_request, SignedHeaders=content-type;host;x-amz-date,
 Signature=4ac9bd4574a14e01568134fd16814534d9951649d3a22b3b0db9f1f5cd4dd0ac
Cache-Control: no-cache
Postman-Token: 9c43020a-966f-61e1-81af-4c49ad8d1392
```

Si todo sale bien, debe recibir una respuesta `200 OK` con una carga XML que muestre un único elemento en el bucket especificado, a menos que haya añadido más archivos al bucket antes de enviar esta solicitud.

```
<?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**
>
> Para cargar o descargar una imagen, debe configurar el tratamiento de contenido como CONVERT_TO_BINARY.

Tutorial: Creación de una API de REST como proxy de Amazon Kinesis

Esta página describe cómo crear y configurar una API de REST con una integración del tipo `AWS` para tener acceso a Kinesis.

> **Note**
>
> Para integrar la API de API Gateway con Kinesis, debe elegir una región en la que estén disponibles los servicios API Gateway y Kinesis. Para conocer la disponibilidad de las regiones, consulte Puntos de conexión y cuotas de servicio.

A modo de ejemplo, crearemos una API de ejemplo para permitir que un cliente haga lo siguiente:

1. Listado de los flujos disponibles para el usuario en Kinesis
2. Creación, descripción o eliminación de un flujo especificado
3. Leer registros de datos o escribir registros de datos en un flujo especificado

Para realizar las tareas anteriores, la API expone métodos en diferentes recursos para invocar lo siguiente, respectivamente:

1. La acción `ListStreams` en Kinesis
2. La acción `CreateStream`, `DescribeStream` o `DeleteStream`
3. La acción `GetRecords` o `PutRecords` (incluida la acción `PutRecord`) en Kinesis

En concreto, crearemos la API mediante las operaciones siguientes:

- Exponer un método HTTP GET en el recurso `/streams` de la API e integrar el método con la acción ListStreams de Kinesis para mostrar los flujos de la cuenta del intermediario.
- Exponer un método HTTP POST en el recurso `/streams/{stream-name}` de la API e integrar el método con la acción CreateStream de Kinesis para crear un flujo con nombre en la cuenta del intermediario.
- Exponer un método HTTP GET en el recurso `/streams/{stream-name}` de la API e integrar el método con la acción DescribeStream de Kinesis para describir un flujo con nombre en la cuenta del intermediario.
- Exponer un método HTTP DELETE en el recurso `/streams/{stream-name}` de la API e integrar el método con la acción DeleteStream de Kinesis para eliminar un flujo en la cuenta del intermediario.
- Exponer un método PUT HTTP en el recurso `/streams/{stream-name}/record` de la API e integrar el método con la acción PutRecord de Kinesis. Esto permite que el cliente añada un solo registro de datos al flujo con nombre.
- Exponer un método PUT HTTP en el recurso `/streams/{stream-name}/records` de la API e integrar el método con la acción PutRecords de Kinesis. Esto permite que el cliente añada una lista de registros de datos al flujo con nombre.
- Exponer un método GET HTTP en el recurso `/streams/{stream-name}/records` de la API e integrar el método con la acción GetRecords de Kinesis. Esto permite que el cliente muestre registros de datos en el flujo con nombre, con un iterador de fragmento especificado. Un iterador de fragmento especifica la posición del fragmento desde la que se empiezan a leer los registros de datos de forma secuencial.
- Exponer un método GET HTTP en el recurso `/streams/{stream-name}/sharditerator` de la API e integrar el método con la acción GetShardIterator de Kinesis. Este método auxiliar debe proporcionarse en la acción `ListStreams` de Kinesis.

Puede aplicar las instrucciones que se presentan aquí a otras acciones de Kinesis. Para obtener la lista completa de las acciones de Kinesis, consulte Referencia de API de Amazon Kinesis.

En lugar de utilizar la consola de API Gateway para crear la API de ejemplo, puede importar la API de ejemplo en API Gateway mediante la API de importación de API Gateway. Para obtener información acerca de cómo utilizar Import API, consulte Desarrollo de API de REST mediante OpenAPI en API Gateway.

Creación de un rol y una política de IAM para que la API tenga acceso a Kinesis

Para que la API pueda invocar las acciones de Kinesis necesarias, debe tener las políticas de IAM adecuadas asociadas a un rol de IAM. En este paso, deberá crear un nuevo rol de IAM.

Para crear el rol de ejecución del proxy de servicio de AWS

1. Inicie sesión en Consola de administración de AWS y abra la consola IAM en https://console.aws.amazon.com/iam/.
2. Elija Roles.
3. Elija Crear rol.
4. Elija Servicio de AWS en Seleccionar el tipo de entidad de confianza y, a continuación, elija API Gateway y seleccione Permite que API Gateway envíe registros a CloudWatch Logs.
5. Seleccione Siguiente y de nuevo Siguiente.
6. En Nombre de rol, escriba **`APIGatewayKinesisProxyPolicy`** y luego elija Crear rol.
7. En la lista Roles, elija el rol que acaba de crear. Puede que tenga que desplazarse o usar la barra de búsqueda para encontrar el rol.
8. Para el rol seleccionado, seleccione la pestaña Agregar permisos.
9. Elija Adjuntar políticas en la lista desplegable.
10. En la barra de búsqueda, escriba **`AmazonKinesisFullAccess`** y, a continuación, elija Añadir permisos.

    > **Note**
    >
    > Para simplificar el proceso, este tutorial utiliza una política administrada. Como práctica recomendada, se deben crear políticas de IAM propias para otorgar los permisos mínimos requeridos.

11. Anote el ARN del rol recién creado, ya que lo usará más adelante.

Creación de una API como un proxy de Kinesis

Utilice los siguientes pasos para crear la API en la consola de API Gateway.

Para crear una API como un proxy de servicio de AWS para Kinesis

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Si es la primera vez que utiliza API Gateway, verá una página en la que aparecen las características del servicio. En REST API, elija Build (Compilación). Cuando aparezca el menú emergente Create Example API (Crear API de ejemplo), elija OK (Aceptar).

   Si esta no es la primera vez que utiliza API Gateway, elija Create API (Crear API). En REST API, elija Build (Compilación).
3. Elija New API (Nueva API).
4. En API name (Nombre de la API), escriba **`KinesisProxy`**. Mantenga los valores predeterminados en todos los demás campos.
5. (Opcional) En Description (Descripción), introduzca una descripción.
6. En Tipo de dirección IP, seleccione IPv4.
7. Seleccione Create API (Crear API).

Una vez creada la API, la consola de API Gateway mostrará la página Resources (Recursos), que contiene únicamente el recurso raíz de la API (`/`).

Lista de flujos en Kinesis

Kinesis es compatible con la acción `ListStreams` con la siguiente llamada a la API de REST:

```
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>

{
   ...
}
```

En la anterior solicitud de API de REST, la acción se especifica en el parámetro de consulta `Action`. También puede, en su lugar, especificar la acción en un encabezado `X-Amz-Target`:

```
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
{
   ...
}
```

En este tutorial usamos el parámetro de consulta para especificar la acción.

Para exponer una acción de Kinesis en la API, agregue un recurso `/streams` a la raíz de la API. A continuación, defina un método `GET` en el recurso e integre el método en la acción `ListStreams` de Kinesis.

En el siguiente procedimiento se describe cómo listar los flujos de Kinesis con la consola de API Gateway.

Para listar los flujos de Kinesis con la consola de API Gateway

1. Seleccione el recurso `/` y, a continuación, elija Crear recurso.
2. En Nombre del recurso, escriba **`streams`**.
3. Mantenga desactivado CORS (uso compartido de recursos entre orígenes).
4. Elija Crear recurso.
5. Elija el recurso `/streams` y luego Crear método. A continuación, haga lo siguiente:

   a. En Tipo de método, seleccione GET.

   > **Note**
   >
   > El verbo HTTP de un método invocado por un cliente puede ser diferente del verbo HTTP de una integración requerida por el backend. Aquí elegimos `GET` porque mostrar los flujos es intuitivamente una operación READ.

b. En Tipo de integración, seleccione Servicio de AWS.

c. En Región de AWS, seleccione la Región de AWS donde creó el flujo de Kinesis.

d. En Servicio de AWS, seleccione Kinesis.

e. Deje Subdominio de AWS en blanco.

f. En Método HTTP, elija POST.

> **Note**
>
> Aquí elegimos `POST` porque Kinesis requiere que la acción `ListStreams` también se invoque.

g. En Tipo de acción, elija Usar nombre de acción.

h. En Nombre de la función, introduzca **`ListStreams`**.

i. En Rol de ejecución, escriba el ARN del rol de ejecución.

j. Deje el valor predeterminado Acceso directo en Tratamiento de contenido.

k. Elija Crear método.

6. En la pestaña Solicitud de integración, en Configuración de la solicitud de integración, seleccione Editar.

7. En Acceso directo de cuerpo de la solicitud, elija Cuando no haya plantillas definidas (recomendado).

8. Elija Parámetros de encabezados de consulta de URL y luego haga lo siguiente:

   a. Seleccione Añadir parámetros de encabezados de solicitud.

   b. En Nombre, escriba **`Content-Type`**.

   c. En Asignado desde, introduzca **`'application/x-amz-json-1.1'`**.

   Hemos utilizado un mapeo de parámetros de solicitud para definir el encabezado `Content-Type` en el valor estático `'application/x-amz-json-1.1'` para informar a Kinesis de que la entrada es de una versión específica de JSON.

9. Elija Plantillas de mapeo y, a continuación, elija Agregar plantilla de mapeo y haga lo siguiente:

   a. En Content-Type, indique **`application/json`**.

   b. En Cuerpo de plantilla, escriba **`{}`**.

   c. Seleccione Save.

La solicitud ListStreams toma una carga con el siguiente formato JSON:

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

Sin embargo, las propiedades son opcionales. Para utilizar los valores predeterminados, usamos aquí una carga JSON vacía.

10. Pruebe el método GET en el recurso /streams para invocar la acción `ListStreams` en Kinesis:

    Elija la pestaña Prueba. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña.

    Elija Prueba para probar su método.

    Si ya ha creado dos flujos llamados "myStream" y "yourStream" en Kinesis, la prueba realizada correctamente devolverá una respuesta 200 OK que contiene la siguiente carga:

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

Creación, descripción y eliminación de un flujo en Kinesis

Crear, describir y eliminar un flujo en Kinesis implica realizar las siguientes solicitudes a la API de REST de Kinesis, respectivamente:

```
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"
}
```

Podemos crear la API para que acepte la entrada necesaria como una carga JSON de una solicitud de método y transmita la carga a la solicitud de integración. Sin embargo, para proporcionar más ejemplos de mapeo de datos entre solicitudes de método e integración y respuestas de método e integración, crearemos nuestra API de una forma ligeramente distinta.

Exponemos los métodos HTTP `GET`, `POST` y `Delete` HTTP en un recurso `Stream` al que se asignará un nombre. Utilizamos la variable de ruta `{stream-name}` como marcador de posición

del recurso de flujo e integramos estos métodos de la API con las acciones `DescribeStream`, `CreateStream` y `DeleteStream` de Kinesis respectivamente. Exigimos que el cliente pase otros datos de entrada como encabezados, parámetros de consulta o la carga de una solicitud de método. Proporcionamos plantillas de asignación para transformar los datos en la carga de solicitud de integración necesaria.

Para crear un recurso {stream-name}

1. Elija el recurso /streams y, a continuación, elija Crear recurso.
2. Mantenga Recurso proxy desactivado.
3. En Ruta de recurso, seleccione `/streams`.
4. En Nombre del recurso, escriba **`{stream-name}`**.
5. Mantenga desactivado CORS (uso compartido de recursos entre orígenes).
6. Elija Crear recurso.

Para configurar y probar el método GET en un recurso de flujo

1. Elija el recurso /{stream-name} y, a continuación, elija Crear método.
2. En Tipo de método, seleccione GET.
3. En Tipo de integración, seleccione Servicio de AWS.
4. En Región de AWS, seleccione la Región de AWS donde creó el flujo de Kinesis.
5. En Servicio de AWS, seleccione Kinesis.
6. Deje Subdominio de AWS en blanco.
7. En Método HTTP, elija POST.
8. En Tipo de acción, elija Usar nombre de acción.
9. En Nombre de la función, introduzca **`DescribeStream`**.
10. En Rol de ejecución, escriba el ARN del rol de ejecución.
11. Deje el valor predeterminado Acceso directo en Tratamiento de contenido.
12. Elija Crear método.
13. En la sección Solicitud de integración, añada los siguientes Parámetros de encabezados de solicitudes de URL:

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

La tarea sigue el mismo procedimiento para configurar el mapeo del parámetro de solicitud que el método `GET /streams`.

14. Añada la siguiente plantilla de mapeo de cuerpo para mapear los datos de la solicitud del método `GET /streams/{stream-name}` a la solicitud de integración `POST /?Action=DescribeStream`:

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

Esta plantilla de mapeo genera la carga de la solicitud de integración necesaria para la acción `DescribeStream` de Kinesis a partir del valor del parámetro de ruta `stream-name` de la solicitud de método.

15. Para probar el método `GET /stream/{stream-name}` para invocar la acción `DescribeStream` en Kinesis, seleccione la pestaña Prueba.
16. En Ruta, en stream-name, introduzca el nombre de un flujo de Kinesis existente.
17. Seleccione Probar Si la prueba tiene éxito, se devuelve la respuesta 200 OK con una carga similar a la siguiente:

```
{
  "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"
  }
}
```

Después de implementar la API, puede realizar una solicitud REST a este método de la API:

```
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
```

Para configurar y probar el método POST en un recurso de flujo

1. Elija el recurso /{stream-name} y, a continuación, elija Crear método.
2. En Tipo de método, seleccione POST.
3. En Tipo de integración, seleccione Servicio de AWS.
4. En Región de AWS, seleccione la Región de AWS donde creó el flujo de Kinesis.
5. En Servicio de AWS, seleccione Kinesis.
6. Deje Subdominio de AWS en blanco.
7. En Método HTTP, elija POST.
8. En Tipo de acción, elija Usar nombre de acción.
9. En Nombre de la función, introduzca **`CreateStream`**.

10. En Rol de ejecución, escriba el ARN del rol de ejecución.
11. Deje el valor predeterminado Acceso directo en Tratamiento de contenido.
12. Elija Crear método.
13. En la sección Solicitud de integración, añada los siguientes Parámetros de encabezados de solicitudes de URL:

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

    La tarea sigue el mismo procedimiento para configurar el mapeo del parámetro de solicitud que el método `GET /streams`.

14. Añada la siguiente plantilla de mapeo de cuerpo para mapear los datos de la solicitud del método `POST /streams/{stream-name}` a la solicitud de integración `POST /?Action=CreateStream`:

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

    En la plantilla de mapeo anterior, definimos `ShardCount` en un valor fijo de 5 si el cliente no especifica un valor en la carga de la solicitud de método.

15. Para probar el método `POST /stream/{stream-name}` para invocar la acción `CreateStream` en Kinesis, seleccione la pestaña Prueba.
16. En Ruta, en stream-name, introduzca el nombre de un flujo de Kinesis nuevo.
17. Seleccione Probar Si la prueba tiene éxito, se devuelve una respuesta 200 OK sin datos.

    Después de implementar la API, también puede realizar una solicitud de API de REST al método POST en un recurso Stream para invocar la acción `CreateStream` en Kinesis:

```
POST https://your-api-id.execute-api.region.amazonaws.com/stage/streams/yourStream
 HTTP/1.1
Host: your-api-id.execute-api.region.amazonaws.com
Content-Type: application/json
Authorization: ...
X-Amz-Date: 20160323T194451Z
```

```
{
    "ShardCount": 5
}
```

Configurar y probar el método DELETE en un recurso de flujo

1. Elija el recurso /{stream-name} y, a continuación, elija Crear método.
2. En Tipo de método, seleccione DELETE.
3. En Tipo de integración, seleccione Servicio de AWS.
4. En Región de AWS, seleccione la Región de AWS donde creó el flujo de Kinesis.
5. En Servicio de AWS, seleccione Kinesis.
6. Deje Subdominio de AWS en blanco.
7. En Método HTTP, elija POST.
8. En Tipo de acción, elija Usar nombre de acción.
9. En Nombre de la función, introduzca **`DeleteStream`**.
10. En Rol de ejecución, escriba el ARN del rol de ejecución.
11. Deje el valor predeterminado Acceso directo en Tratamiento de contenido.
12. Elija Crear método.
13. En la sección Solicitud de integración, añada los siguientes Parámetros de encabezados de solicitudes de URL:

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

    La tarea sigue el mismo procedimiento para configurar el mapeo del parámetro de solicitud que el método `GET /streams`.
14. Añada la siguiente plantilla de mapeo de cuerpo para mapear los datos de la solicitud de método `DELETE /streams/{stream-name}` a la solicitud de integración de `POST /?Action=DeleteStream` correspondiente:

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

Esta plantilla de mapeo genera la entrada necesaria de la `DELETE /streams/{stream-name}` acción desde el nombre de ruta URL proporcionada por el cliente: `stream-name`.

15. Para probar el método `DELETE /stream/{stream-name}` para invocar la acción `DeleteStream` en Kinesis, seleccione la pestaña Prueba.
16. En Ruta, en stream-name, introduzca el nombre de un flujo de Kinesis existente.
17. Seleccione Probar Si la prueba tiene éxito, se devuelve una respuesta 200 OK sin datos.

Después de implementar la API, también puede realizar la siguiente solicitud API de REST al método DELETE en el recurso Stream para llamar a la acción `DeleteStream` en Kinesis:

```
DELETE https://your-api-id.execute-api.region.amazonaws.com/stage/
streams/yourStream HTTP/1.1
Host: your-api-id.execute-api.region.amazonaws.com
Content-Type: application/json
Authorization: ...
X-Amz-Date: 20160323T194451Z

{}
```

Obtención de registros y adición de registros a un flujo de Kinesis

Después de crear un flujo en Kinesis, puede agregar registros de datos al flujo y leer los datos del flujo. Agregar registros de datos implica llamar a la acción PutRecords o PutRecord de Kinesis. La primera acción añade varios registros, mientras que la última añade un solo registro al flujo.

```
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"
}
```

o

```
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"
}
```

Aquí, `StreamName` identifica el flujo de destino para añadir registros. `StreamName`, `Data` y `PartitionKey` son datos de entrada necesarios. En nuestro ejemplo, usamos los valores predeterminados para todos los datos de entrada opcionales y no especificaremos de forma explícita los valores para ellos en la entrada de la solicitud del método.

Leer datos en Kinesis equivale a llamar a la acción GetRecords:

```
POST /?Action=GetRecords HTTP/1.1
Host: kinesis.region.domain
Authorization: AWS4-HMAC-SHA256 Credential=..., ...
...
Content-Type: application/x-amz-json-1.1
Content-Length: PayloadSizeBytes

{
```

```
    "ShardIterator": "string",
    "Limit": number
}
```

Aquí, el flujo de origen del que se obtienen los registros se especifica en el valor de `ShardIterator` necesario, tal y como se muestra en la siguiente acción de Kinesis para obtener un iterador de fragmento:

```
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"
}
```

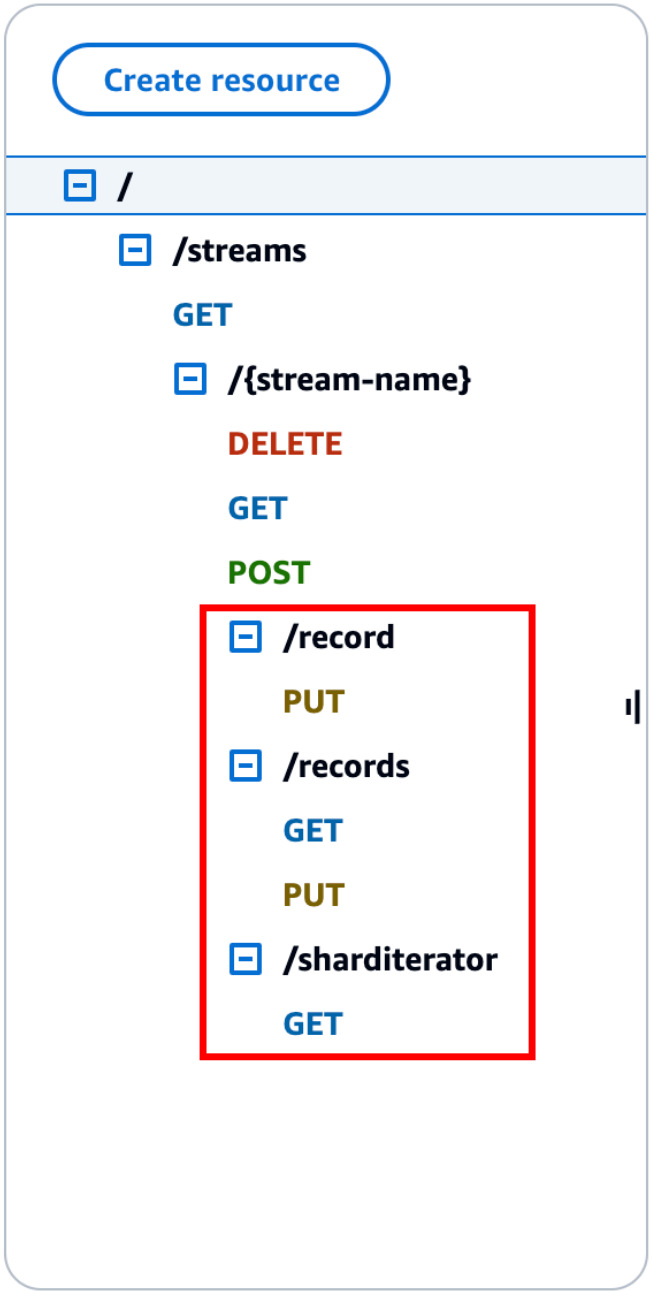
Para las acciones `GetRecords` y `PutRecords`, exponemos los métodos `GET` y `PUT`, respectivamente, en un recurso `/records` que se añade a un recurso de flujo con nombre (`/{stream-name}`). Del mismo modo, exponemos la acción `PutRecord` como un método `PUT` en un recurso `/record`.

Como la acción `GetRecords` toma como entrada un valor de `ShardIterator`, que se obtiene llamando a la acción auxiliar `GetShardIterator`, exponemos un método auxiliar `GET` en un recurso `ShardIterator` (`/sharditerator`).

Para crear los recursos /record, /records y /sharditerator

1. Elija el recurso /{stream-name} y, a continuación, elija Crear recurso.
2. Mantenga Recurso proxy desactivado.
3. En Ruta de recurso, seleccione `/{stream-name}`.
4. En Nombre del recurso, escriba **`record`**.

5. Mantenga desactivado CORS (uso compartido de recursos entre orígenes).
6. Elija Crear recurso.
7. Repita los pasos anteriores para crear un recurso /records y un recurso /sharditerator. La API final debe ser similar a la siguiente:



Los siguientes cuatro procedimientos describen cómo configurar cada uno de los métodos, cómo asignar datos desde las solicitudes de método a las solicitudes de integración y cómo probar los métodos.

Para configurar y probar el método **`PUT /streams/{stream-name}/record`** para invocar a **`PutRecord`** en Kinesis

1. Elija /record y, a continuación, elija Crear método.
2. En Tipo de método, seleccione PUT.
3. En Tipo de integración, seleccione Servicio de AWS.
4. En Región de AWS, seleccione la Región de AWS donde creó el flujo de Kinesis.
5. En Servicio de AWS, seleccione Kinesis.
6. Deje Subdominio de AWS en blanco.
7. En Método HTTP, elija POST.
8. En Tipo de acción, elija Usar nombre de acción.
9. En Nombre de la función, introduzca **`PutRecord`**.
10. En Rol de ejecución, escriba el ARN del rol de ejecución.
11. Deje el valor predeterminado Acceso directo en Tratamiento de contenido.
12. Elija Crear método.
13. En la sección Solicitud de integración, añada los siguientes Parámetros de encabezados de solicitudes de URL:

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

    La tarea sigue el mismo procedimiento para configurar el mapeo del parámetro de solicitud que el método `GET /streams`.
14. Añada la siguiente plantilla de mapeo de cuerpo para mapear los datos de la solicitud de método `PUT /streams/{stream-name}/record` a la solicitud de integración de `POST /?Action=PutRecord` correspondiente:

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

En esta plantilla de asignación se asume que la carga del método de solicitud tiene el siguiente formato:

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

Estos datos se pueden modelar mediante el siguiente esquema JSON:

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

Puede crear un modelo para incluir este esquema y utilizar el modelo para facilitar la generación de la plantilla de asignación. Sin embargo, puede generar una plantilla de asignación sin usar ningún modelo.

15. Para probar el método `PUT /streams/{stream-name}/record`, establezca la variable de ruta `stream-name` en el nombre de un flujo existente, introduzca la carga del formato necesario y, a continuación, envíe la solicitud de método. El resultado correcto es una respuesta `200 OK` con una carga con el formato siguiente:

```
{
  "SequenceNumber": "49559409944537880850133345460169886593573102115167928386",
  "ShardId": "shardId-000000000004"
}
```

Para configurar y probar el método **`PUT /streams/{stream-name}/records`** para invocar a **`PutRecords`** en Kinesis

1. Elija el recurso /records y, a continuación, elija Crear método.
2. En Tipo de método, seleccione PUT.
3. En Tipo de integración, seleccione Servicio de AWS.
4. En Región de AWS, seleccione la Región de AWS donde creó el flujo de Kinesis.
5. En Servicio de AWS, seleccione Kinesis.
6. Deje Subdominio de AWS en blanco.
7. En Método HTTP, elija POST.
8. En Tipo de acción, elija Usar nombre de acción.
9. En Nombre de la función, introduzca **`PutRecords`**.
10. En Rol de ejecución, escriba el ARN del rol de ejecución.
11. Deje el valor predeterminado Acceso directo en Tratamiento de contenido.
12. Elija Crear método.
13. En la sección Solicitud de integración, añada los siguientes Parámetros de encabezados de solicitudes de URL:

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

    La tarea sigue el mismo procedimiento para configurar el mapeo del parámetro de solicitud que el método `GET /streams`.
14. Añada la siguiente plantilla de mapeo para mapear los datos de la solicitud del método `PUT /streams/{stream-name}/records` a la solicitud de integración de `POST /?Action=PutRecords` correspondiente:

    ```
    {
        "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
        ]
    ```

```
}
```

Esta plantilla de mapeo asume que el siguiente esquema JSON puede modelar la carga de la solicitud de método:

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

Puede crear un modelo para incluir este esquema y utilizar el modelo para facilitar la generación de la plantilla de asignación. Sin embargo, puede generar una plantilla de asignación sin usar ningún modelo.

En este tutorial, hemos usado dos formatos de carga ligeramente diferentes para ilustrar que un desarrollador de API puede optar por exponer el formato de datos de backend al cliente u ocultárselo. Un formato es para el método `PUT /streams/{stream-name}/records` (anterior). Otro formato se utiliza en el método `PUT /streams/{stream-name}/record` (en el procedimiento anterior). En un entorno de producción, debe mantener la coherencia de ambos formatos.

15. Para probar el método `PUT /streams/{stream-name}/records`, establezca la variable de ruta `stream-name` en un flujo existente, introduzca la siguiente carga y envíe la solicitud del método.

```
{
```

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

El resultado correcto es una respuesta 200 OK con una carga similar a la siguiente:

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

Para configurar y probar el método **`GET /streams/{stream-name}/sharditerator`** para invocar a **`GetShardIterator`** en Kinesis

`GET /streams/{stream-name}/sharditerator` es un método auxiliar para adquirir un iterador de fragmento necesario antes de llamar al método `GET /streams/{stream-name}/records`.

1. Elija el recurso /sharditerator y, a continuación, elija Crear método.
2. En Tipo de método, seleccione GET.
3. En Tipo de integración, seleccione Servicio de AWS.
4. En Región de AWS, seleccione la Región de AWS donde creó el flujo de Kinesis.
5. En Servicio de AWS, seleccione Kinesis.

6. Deje Subdominio de AWS en blanco.
7. En Método HTTP, elija POST.
8. En Tipo de acción, elija Usar nombre de acción.
9. En Nombre de la función, introduzca **`GetShardIterator`**.
10. En Rol de ejecución, escriba el ARN del rol de ejecución.
11. Deje el valor predeterminado Acceso directo en Tratamiento de contenido.
12. Elija Parámetros de cadenas de consulta de URL.

    La acción `GetShardIterator` requiere una entrada de un valor de ShardId. Para pasar un valor de `ShardId` proporcionado por el cliente, añadimos un parámetro de consulta `shard-id` a la solicitud del método, tal y como se muestra en el siguiente paso.
13. Elija Añadir cadena de consulta.
14. En Nombre, escriba **`shard-id`**.
15. Mantenga desactivados Obligatorio y Almacenamiento en caché.
16. Elija Crear método.
17. En la sección Solicitud de integración, añada la siguiente plantilla de mapeo para generar la entrada requerida (`ShardId` y `StreamName`) a la acción `GetShardIterator` a partir de los parámetros `shard-id` y `stream-name` de la solicitud de método. Además, la plantilla de mapeo también establece `ShardIteratorType` en `TRIM_HORIZON` como valor predeterminada.

    ```
    {
        "ShardId": "$input.params('shard-id')",
        "ShardIteratorType": "TRIM_HORIZON",
        "StreamName": "$input.params('stream-name')"
    }
    ```
18. Con la opción Test (Prueba) de la consola de API Gateway, escriba un nombre de flujo existente como el valor de la variable `stream-name` de Path (Ruta), establezca `shard-id` para Query string (Cadena de consulta) en un valor `ShardId` existente (por ejemplo, `shard-000000000004`) y elija Test (Prueba).

    La carga de la respuesta correcta es similar al siguiente resultado:

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

```
}
```

Anote el valor de `ShardIterator`. Lo necesita para obtener los registros de un flujo.

Para configurar y probar el método **`GET /streams/{stream-name}/records`** para invocar la acción **`GetRecords`** en Kinesis

1. Elija el recurso /records y, a continuación, elija Crear método.
2. En Tipo de método, seleccione GET.
3. En Tipo de integración, seleccione Servicio de AWS.
4. En Región de AWS, seleccione la Región de AWS donde creó el flujo de Kinesis.
5. En Servicio de AWS, seleccione Kinesis.
6. Deje Subdominio de AWS en blanco.
7. En Método HTTP, elija POST.
8. En Tipo de acción, elija Usar nombre de acción.
9. En Nombre de la función, introduzca **`GetRecords`**.
10. En Rol de ejecución, escriba el ARN del rol de ejecución.
11. Deje el valor predeterminado Acceso directo en Tratamiento de contenido.
12. Elija Encabezados de solicitud HTTP.

    La acción `GetRecords` requiere una entrada de un valor de `ShardIterator`. Para pasar un valor de `ShardIterator` proporcionado por el cliente, agregamos un parámetro de encabezado `Shard-Iterator` a la solicitud del método.
13. Elija Agregar encabezado.
14. En Nombre, escriba **`Shard-Iterator`**.
15. Mantenga desactivados Obligatorio y Almacenamiento en caché.
16. Elija Crear método.
17. En la sección Solicitud de integración, añada la siguiente plantilla de mapeo de cuerpo para mapear el valor del parámetro de encabezado `Shard-Iterator` al valor de la propiedad `ShardIterator` de la carga JSON para la acción `GetRecords` en Kinesis.

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

18. Con la opción Prueba de la consola de API Gateway, escriba un nombre de flujo existente como el valor de la variable `stream-name` de Ruta, establezca el `Shard-Iterator` de Encabezado en el valor de `ShardIterator` obtenido de la serie de pruebas del método `GET /streams/{stream-name}/sharditerator` (arriba) y elija Prueba.

    La carga de la respuesta correcta es similar al siguiente resultado:

```
{
  "MillisBehindLatest": 0,
  "NextShardIterator": "AAAAAAAAAAF...",
  "Records": [ ... ]
}
```

Definiciones de OpenAPI de la API de ejemplo como un proxy de Kinesis

A continuación se muestran las definiciones de OpenAPI de la API de ejemplo, utilizada en este tutorial, como un proxy de Kinesis.

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: Creación de una API de REST mediante SDK de AWS o la AWS CLI

En el siguiente tutorial se muestra cómo crear una API de PetStore que sea compatible con los métodos `GET /pets` y `GET /pets/{petId}`. Los métodos están integrados con un punto de conexión HTTP. Puede seguir este tutorial con el AWS SDK para JavaScript, el AWS SDK para Python (Boto3) o la AWS CLI. Se utilizan las siguientes funciones o comandos para configurar la API:

JavaScript v3

- CreateRestApiCommand
- CreateResourceCommand
- PutMethodCommand
- PutMethodResponseCommand
- PutIntegrationCommand
- PutIntegrationResponseCommand
- CreateDeploymentCommand

Python

- create_rest_api
- create_resource
- put_method
- put_method_response
- put_integration
- put_integration_response
- create_deployment

AWS CLI

- create-rest-api
- create-resource

- put-method
- put-method-response
- put-integration
- put-integration-response
- create-deployment

Para obtener más información sobre AWS SDK para JavaScript v3, consulte ¿Qué es AWS SDK para JavaScript?. Para obtener más información sobre el SDK para Python (Boto3), consulte AWS SDK para Python (Boto3). Para obtener más información sobre la AWS CLI, consulte ¿Qué es la AWS CLI?.

Configuración de una API PetStore optimizada para sistemas perimetrales

En este tutorial, los comandos de ejemplo utilizan valores de marcador de posición para los ID de valor, como el ID de API y el ID de recurso. Según complete el tutorial, sustituya estos valores por los suyos.

Configuración de una API de PetStore optimizada para sistemas perimetrales con AWS SDK

1. Utilice el siguiente ejemplo para crear una entidad `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)
}
```

```
})();
```

Una llamada correcta devuelve el ID de API y el ID de recurso raíz de la API en un resultado como el siguiente:

```
{
  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)
```

Una llamada correcta devuelve el ID de API y el ID de recurso raíz de la API en un resultado como el siguiente:

```
{'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
```

A continuación se muestra la salida de este comando:

```
{
    "id": "abcd1234",
    "name": "Simple PetStore (AWS CLI)",
    "createdDate": "2022-12-15T08:07:04-08:00",
    "apiKeySource": "HEADER",
    "endpointConfiguration": {
        "types": [
            "EDGE"
        ]
    },
    "disableExecuteApiEndpoint": false,
    "rootResourceId": "efg567"
}
```

La API que ha creado tiene el ID de API `abcd1234` y el ID de recurso raíz `efg567`. Se utilizan estos valores en la configuración de la API.

2. A continuación, se agrega un recurso secundario en la raíz, se especifica `RootResourceId` como valor de la propiedad `parentId`. Use el siguiente ejemplo para crear un recurso de `/pets` para la API:

   JavaScript v3

```
import {APIGatewayClient,  CreateResourceCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
```

```
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new CreateResourceCommand({
    restApiId: 'abcd1234',
    parentId: 'efg567',
    pathPart: 'pets'
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.log("The '/pets' resource setup failed:\n", err)
}
})();
```

Una llamada correcta devuelve información sobre el recurso en un resultado como el siguiente:

```
{
    "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)
```

Una llamada correcta devuelve información sobre el recurso en un resultado como el siguiente:

```
{'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
```

A continuación se muestra la salida de este comando:

```
{
    "id": "aaa111",
    "parentId": "efg567",
    "pathPart": "pets",
    "path": "/pets"
}
```

El recurso `/pets` que ha creado tiene un ID de recurso de `aaa111`. Se utiliza este valor en la configuración de la API.

3. A continuación, agregue un recurso secundario en el recurso de `/pets`. Este recurso, `/{petId}` tiene un parámetro de ruta para `{petId}`. Para hacer una parte de la ruta, un parámetro de la ruta, sitúelo entre llaves, `{ }`. Use el siguiente ejemplo para crear un recurso de `/pets/{petId}` para la API:

   JavaScript v3

   ```
   import {APIGatewayClient,  CreateResourceCommand } from "@aws-sdk/client-api-
   gateway";
   (async function (){
   const apig = new APIGatewayClient({region:"us-east-1"});
   const command = new CreateResourceCommand({
   ```

```
    restApiId: 'abcd1234',
    parentId: 'aaa111',
    pathPart: '{petId}'
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.log("The '/pets/{petId}' resource setup failed:\n", err)
}
})();
```

Una llamada correcta devuelve información sobre el recurso en un resultado como el siguiente:

```
{
    "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)
```

Una llamada correcta devuelve información sobre el recurso en un resultado como el siguiente:

```
{'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}'
```

A continuación se muestra la salida de este comando:

```
{
    "id": "bbb222",
    "parentId": "aaa111",
    "path": "/pets/{petId}",
    "pathPart": "{petId}"
}
```

El recurso `/pets/{petId}` que ha creado tiene un ID de recurso de `bbb222`. Se utiliza este valor en la configuración de la API.

4. En los dos pasos siguientes, se agregan métodos HTTP a los recursos. En este tutorial, se establecen los métodos para que tengan acceso abierto configurando el `authorization-type` para establecer en `NONE`. Para permitir que solo puedan llamar al método los usuarios autenticados, puede utilizar políticas y roles de IAM, un autorizador de Lambda (que anteriormente se denominaba autorizador personalizado) o un grupo de usuarios de Amazon Cognito. Para obtener más información, consulte the section called “Control de acceso”.

   Use el siguiente ejemplo para agregar el método HTTP `GET` al recurso `/pets`:

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)
}
})();
```

Una llamada correcta devuelve el resultado siguiente:

```
{
    "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)
```

Una llamada correcta devuelve el resultado siguiente:

```
{'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
```

A continuación se muestra la salida de este comando:

```
{
    "httpMethod": "GET",
    "authorizationType": "NONE",
    "apiKeyRequired": false
}
```

5. Use el siguiente ejemplo para agregar el método HTTP `GET` al recurso `/pets/{petId}` y establecer la propiedad `requestParameters` para pasar el valor `petId` proporcionado por el cliente al backend:

   JavaScript v3

```
import {APIGatewayClient,  PutMethodCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new PutMethodCommand({
```

```
    restApiId: 'abcd1234',
    resourceId: 'bbb222',
    httpMethod: 'GET',
    authorizationType: 'NONE'
    requestParameters: {
        "method.request.path.petId" : true
    }
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.log("The 'GET /pets/{petId}' method setup failed:\n", err)
}
})();
```

Una llamada correcta devuelve el resultado siguiente:

```
{
    "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)
```

Una llamada correcta devuelve el resultado siguiente:

```
{'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
```

A continuación se muestra la salida de este comando:

```
{
    "httpMethod": "GET",
    "authorizationType": "NONE",
    "apiKeyRequired": false,
    "requestParameters": {
        "method.request.path.petId": true
    }
}
```

6. Use el siguiente ejemplo para agregar la respuesta del método 200 OK para el método `GET /pets`:

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)
}
})();
```

Una llamada correcta devuelve el resultado siguiente:

```
{
    "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)
```

Una llamada correcta devuelve el resultado siguiente:

```
{'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
```

A continuación se muestra la salida de este comando:

```
{
    "statusCode": "200"
}
```

7. Use el siguiente ejemplo para agregar la respuesta del método 200 OK para el método `GET /pets/{petId}`:

   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)
}
})();
```

Una llamada correcta devuelve el resultado siguiente:

```
{
    "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)
```

Una llamada correcta devuelve el resultado siguiente:

```
{'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
```

A continuación se muestra la salida de este comando:

```
{
    "statusCode": "200"
}
```

8. Use el siguiente ejemplo para configurar una integración para el método `GET /pets` con un punto de conexión HTTP. El punto de conexión HTTPS es `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)
}
})();
```

Una llamada correcta devuelve el resultado siguiente:

```
{
    "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)
```

Una llamada correcta devuelve el resultado siguiente:

```
{'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
```

A continuación se muestra la salida de este comando:

```
{
    "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. Use el siguiente ejemplo para configurar una integración para el método `GET /pets/{petId}` con un punto de conexión HTTP. El punto de conexión HTTPS es `http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}`. En este paso, se mapea el parámetro de ruta `petId` al parámetro de ruta del punto de conexión `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)
}
})();
```

Una llamada correcta devuelve el resultado siguiente:

```
{
    "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)
```

Una llamada correcta devuelve el resultado siguiente:

```
{'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
```

A continuación se muestra la salida de este comando:

```
{
    "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. Use el siguiente ejemplo para agregar la respuesta de integración para la integración de `GET /pets`:

    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)
    }
    })();
    ```

    Una llamada correcta devuelve el resultado siguiente:

    ```
    {
        "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)
```

Una llamada correcta devuelve el resultado siguiente:

```
{'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
```

A continuación se muestra la salida de este comando:

```
{
    "statusCode": "200",
    "selectionPattern": ""
```

```
}
```

11. Use el siguiente ejemplo para agregar la respuesta de integración para la integración de `GET /pets/{petId}`:

    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)
    }
    })();
    ```

    Una llamada correcta devuelve el resultado siguiente:

    ```
    {
        "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)
```

Una llamada correcta devuelve el resultado siguiente:

```
{'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
```

A continuación se muestra la salida de este comando:

```
{
    "statusCode": "200",
    "selectionPattern": ""
}
```

Después de crear la respuesta de integración, la API puede consultar las mascotas disponibles en el sitio web de PetStore y ver una mascota concreta de un identificador especificado. Antes de que los clientes puedan llamar a la API, debe implementarla. Le recomendamos que antes de implementar la API, la pruebe.

12. Use el siguiente ejemplo para probar el método `GET /pets`:

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. Use el siguiente ejemplo para probar el método `GET /pets/{petId}` con un `petId` de 3:

JavaScript v3

```
import {APIGatewayClient,  TestInvokeMethodCommand } from "@aws-sdk/client-api-
gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new TestInvokeMethodCommand({
    restApiId: 'abcd1234',
    resourceId: 'bbb222',
    httpMethod: 'GET',
    pathWithQueryString: '/pets/3',
});
try {
    const results = await apig.send(command)
    console.log(results)
} catch (err) {
    console.log("The test on 'GET /pets/{petId}' method failed:\n", err)
}
})();
```

Python

```
import botocore
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.test_invoke_method(
        restApiId='abcd1234',
        resourceId='bbb222',
```

```
        httpMethod='GET',
        pathWithQueryString='/pets/3',
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Test invoke method on 'GET /pets/{petId}' failed: %s",
 error)
    raise
print(result)
```

AWS CLI

```
aws apigateway test-invoke-method --rest-api-id abcd1234 /
  --resource-id bbb222 /
  --http-method GET /
  --path-with-query-string '/pets/3'
```

Una vez que haya probado con éxito la API, puede implementarla en una etapa.

14. Use el siguiente ejemplo para implementar la API en una etapa denominada `test`. Cuando se implementa la API en una etapa, los intermediarios de la API pueden invocar la API.

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'
```

A continuación se muestra la salida de este comando:

```
{
    "id": "ab1c1d",
    "description": "First deployment",
    "createdDate": "2022-12-15T08:44:13-08:00"
}
```

Los clientes ahora pueden llamar a la API. Puede probar esta API ingresando la URL `https://abcd1234.execute-api.us-west-2.amazonaws.com/test/pets` en un navegador y sustituyendo `abcd1234` por el identificador de la API.

Para ver más ejemplos de cómo crear o actualizar una API mediante los AWS SDK o AWS CLI, consulte Acciones para API Gateway con AWS SDK.

Automatización de la configuración de la API

En lugar de crear la API paso a paso, puede automatizar la creación y la limpieza de recursos de AWS mediante OpenAPI, CloudFormation o Terraform para crear la API.

Definición de OpenAPI 3.0

Puede importar una definición de OpenAPI a API Gateway. Para obtener más información, consulte 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 CloudFormationPlantilla de

Para implementar la plantilla de CloudFormation, consulte Creación de una pila en la consola de AWS CloudFormation.

```
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'
```

Configuración de Terraform

Para obtener más información sobre Terraform, consulte 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]
}

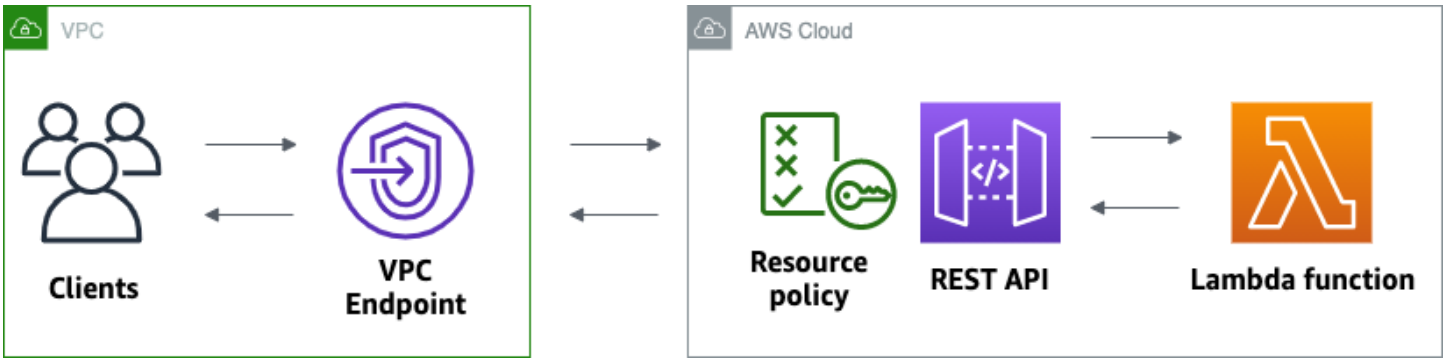
resource "aws_api_gateway_integration_response" "petIdIntegrationResponse" {
    rest_api_id = aws_api_gateway_rest_api.Api.id
    resource_id = aws_api_gateway_resource.petIdResource.id
    http_method = aws_api_gateway_method.petIdMethodGet.http_method
    status_code = aws_api_gateway_method_response.petIdMethodResponseGet.status_code
}


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: Creación de una API de REST privada

En este tutorial, creará una API REST privada. Los clientes solo pueden acceder a la API desde su Amazon VPC La API está aislada de la Internet pública, que es un requisito de seguridad común.

Este tutorial tarda aproximadamente 30 minutos en completarse. En primer lugar, utilice una plantilla de CloudFormation para crear una nube de Amazon VPC, un punto de enlace de la VPC, una función de AWS Lambda y lanzar una instancia de Amazon EC2 que utilizará para probar la API. A continuación, utilice la Consola de administración de AWS para crear una API privada y adjuntar una política de recursos que permita el acceso solo desde el punto de enlace de la VPC. Por último, se prueba la API.



Para completar este tutorial, necesita una cuenta de AWS y un usuario de AWS Identity and Access Management con acceso a la consola. Para obtener más información, consulte Requisitos previos.

En este tutorial, se utiliza la Consola de administración de AWS. Para obtener una plantilla de CloudFormation que cree esta API y todos los recursos relacionados, consulte template.yaml.

Temas

- Paso 1: crear dependencias
- Paso 2: crear una API privada
- Paso 3: crear un método e integración
- Paso 4: adjuntar una política de recursos
- Paso 5: implemente su API
- Paso 6: compruebe que su API no sea accesible públicamente
- Paso 7: conéctese a una instancia de su VPC e invoque su API
- Paso 8: Eliminar
- Próximos pasos: automatice con CloudFormation

Paso 1: crear dependencias

Descargue y descomprima esta plantilla de CloudFormation. Utilice la plantilla para crear todas las dependencias de su API privada, incluidas una VPC de Amazon, un punto de enlace de la VPC y una función Lambda que sirve como backend de la API. Creará la API privada más tarde.

Para crear una pila de CloudFormation

1. Abra la consola de CloudFormation en https://console.aws.amazon.com/cloudformation.
2. Seleccione Create stack (Crear pila) y, a continuación, seleccione With new resources (standard) (Con nuevos recursos [estándar]).
3. En Specify template (Especificar plantilla), elija Upload a template file (Cargar un archivo de plantilla).
4. Seleccione la plantilla que ha descargado.
5. Elija Next (Siguiente).
6. En Stack name (Nombre de pila), escriba **`private-api-tutorial`** y, a continuación, elija Next (Siguiente).
7. En Configure stack options (Configurar opciones de pila), elija Next (Siguiente).
8. Para Capabilities (Capacidades), sepa que CloudFormation puede crear recursos de IAM en su cuenta.
9. Elija Siguiente y, a continuación, elija Enviar.

CloudFormation aprovisiona las dependencias para su API. Puede tardar unos minutos. Cuando el estado de la pila de CloudFormation sea CREATE_COMPLETE, elija Outputs (Salidas). Anote el ID del punto de enlace de la VPC. Los va a necesitar en pasos más adelante de este tutorial.

Paso 2: crear una API privada

Cree una API privada para permitir que solo los clientes de su VPC accedan a ella.

Creación de una API privada

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Seleccione Create API (Crear API)y, a continuación, para API REST, seleccione Build (Construir).
3. En API name (Nombre de la API), escriba**`private-api-tutorial`**.

4. En Tipo de punto de conexión de la API, seleccione Privado.
5. Para ID del punto de conexión de la VPC, introduzca el ID de punto de conexión de la VPC en las Salidas de la pila de CloudFormation.
6. Para Tipo de dirección IP, elija Dualstack.
7. Seleccione Crear API.

Paso 3: crear un método e integración

Usted crea un método y la integración de Lambda `GET` para manejar solicitudes de `GET` a su API. Cuando un cliente invoca su API, API Gateway envía la solicitud a la función Lambda que creó en el paso 1 y, a continuación, devuelve una respuesta al cliente.

Para crear un método e integración

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija la API.
3. Elija Crear método.
4. En Tipo de método, seleccione `GET`.
5. En Tipo de integración, seleccione Función de Lambda.
6. Active Integración de proxy de Lambda. Con una integración de proxy de Lambda, API Gateway envía un evento a Lambda con una estructura definida y transforma la respuesta de su función Lambda a una respuesta HTTP.
7. En la Lambda function (Función de Lambda), elija la función que creó con la plantilla de CloudFormation en el paso 1. El nombre de la función comienza con **`private-api-tutorial`**.
8. Elija Crear método.

Paso 4: adjuntar una política de recursos

Adjuntar una política de recursos a la API que permite a los clientes invocar la API sólo a través del punto de enlace de la VPC. Para restringir aún más el acceso a la API, también puede configurar una política de punto de enlace de la VPC para el punto de enlace de su VPC, pero esto no es necesario para este tutorial.

Para adjuntar una política de recursos

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.

2. Elija la API.
3. Elija Política de recursos y, a continuación, elija Crear política.
4. Escriba la siguiente política. Reemplace *vpceID* por su ID de punto de enlace de la VPC desde las Outputs (Salidas) de su pila de CloudFormation.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "execute-api:/*",
            "Condition": {
                "StringNotEquals": {
                    "aws:sourceVpce": "vpce-abcd1234"
                }
            }
        },
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "execute-api:/*"
        }
    ]
}
```

5. Seleccione Save changes (Guardar cambios).

Paso 5: implemente su API

A continuación, implemente su API para ponerla a disposición de los clientes de su Amazon VPC

Para implementar una API

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija la API.

3. Elija Implementar API.
4. En Etapa, seleccione Nueva etapa.
5. En Stage name (Nombre de etapa), escriba **`test`**.
6. (Opcional) En Description (Descripción), introduzca una descripción.
7. Elija Deploy (Implementar).

Ya está listo para probar su API.

Paso 6: compruebe que su API no sea accesible públicamente

Use `curl` para verificar que no puede invocar la API desde fuera de su Amazon VPC

Para probar la API

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija la API.
3. En el panel de navegación principal, elija Etapas y, a continuación, elija la etapa test.
4. En Detalles de la etapa, elija el icono de copia para copiar la URL de invocación de la API. La URL se ve así `https://abcdef123.execute-api.us-west-2.amazonaws.com/test`. El punto de enlace de la VPC que creó en el paso 1 tiene el DNS privado habilitado, por lo que puede utilizar la URL proporcionada para invocar la API.
5. Utilice curl para intentar invocar la API desde fuera de su VPC.

   ```
   curl https://abcdef123.execute-api.us-west-2.amazonaws.com/test
   ```

   Curl indica que el punto de enlace de su API no se puede resolver. Si obtiene una respuesta diferente, vuelva al paso 2 y asegúrese de elegir Private (Privado) para el tipo de punto de enlace de su API.

   ```
   curl: (6) Could not resolve host: abcdef123.execute-api.us-west-2.amazonaws.com/
   test
   ```

A continuación, se conecta a una instancia de Amazon EC2 en la VPC para invocar su API.

Paso 7: conéctese a una instancia de su VPC e invoque su API

A continuación, pruebe la API desde su Amazon VPC Para acceder a su API privada, se conecta a una instancia de Amazon EC2 en su VPC y, a continuación, utilice curl para invocar su API. Utilice el Administrador de sesiones de Systems Manager para conectarse a la instancia en el navegador.

Para probar la API

1. Abra la consola de Amazon EC2 en https://console.aws.amazon.com/ec2/.
2. Elija Instances.
3. Elija la instancia denominada private-api-tutorial que creó con la plantilla de CloudFormation en el paso 1.
4. Elija Connect (Conectar) y, a continuación, elija Session Manager (Administrador de sesiones).
5. Elija Connect (Conectar) para iniciar una sesión basada en explorador en la instancia.
6. En su sesión de Session Manager, use curl para invocar su API. Puede invocar la API porque está utilizando una instancia en su Amazon VPC

   ```
   curl https://abcdef123.execute-api.us-west-2.amazonaws.com/test
   ```

   Verifique que obtiene la respuesta `Hello from Lambda!`.



Ha creado correctamente una API a la que solo se puede acceder desde su VPC de Amazon y, a continuación, verificó que funciona.

Paso 8: Eliminar

Para evitar costos innecesarios, elimine los recursos creados como parte de este tutorial Los siguientes pasos eliminan su API REST y su pila de CloudFormation.

Para eliminar una API REST

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. En la página API, seleccione una API. Elija Acciones de API, elija Eliminar API y confirme su elección.

Para eliminar una pila de CloudFormation

1. Abra la consola de CloudFormation en https://console.aws.amazon.com/cloudformation.
2. Seleccione su pila de CloudFormation.
3. Elija Delete (Eliminar) y, a continuación, confirme su elección.

Próximos pasos: automatice con CloudFormation

Puede automatizar la creación y la limpieza de todos los recursos de AWS involucrados en este tutorial. Para obtener una plantilla de CloudFormation de ejemplo completa, consulte template.yaml.

Tutoriales sobre API HTTP de Amazon API Gateway

Los siguientes tutoriales proporcionan ejercicios prácticos para ayudar a saber más sobre las API HTTP de API Gateway.

Temas

- Tutorial: Creación de una API de HTTP CRUD con Lambda y DynamoDB
- Tutorial: Creación de una API de HTTP con una integración privada en un servicio de Amazon ECS

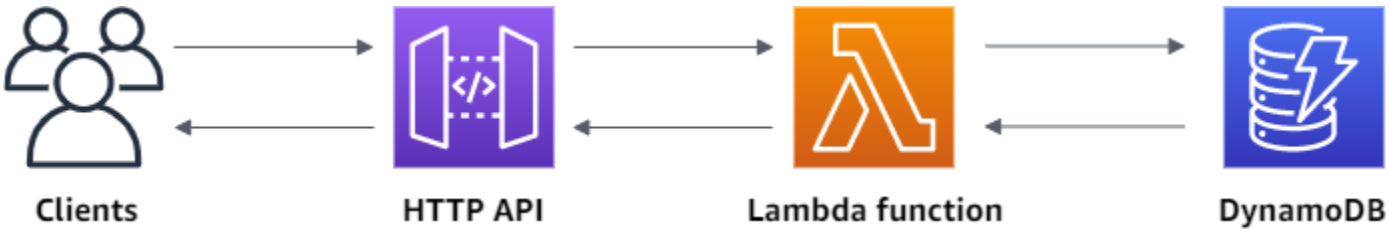
Tutorial: Creación de una API de HTTP CRUD con Lambda y DynamoDB

En este tutorial, creará una API sin servidor para crear, leer, actualizar y eliminar elementos de una tabla de DynamoDB. DynamoDB es un servicio de bases de datos NoSQL totalmente administrado

que proporciona un rendimiento rápido y predecible, así como una perfecta escalabilidad. Completar este tutorial lleva aproximadamente 30 minutos, y puede hacerse dentro del nivel gratuito de AWS.

En primer lugar, se crea una tabla DynamoDB utilizando la consola de DynamoDB. Luego, se crea una función de Lambda con la consola de AWS Lambda. A continuación, crea una API HTTP mediante la consola de API Gateway. Por último, se prueba la API.

Cuando invoca su API HTTP, API Gateway enruta la solicitud a su función de Lambda. La función Lambda interactúa con DynamoDB y devuelve una respuesta a API Gateway. API Gateway, a continuación, le devuelve una respuesta.



Para completar este ejercicio, necesita una cuenta de AWS y un usuario de AWS Identity and Access Management con acceso a la consola. Para obtener más información, consulte Requisitos previos.

En este tutorial, se utiliza la Consola de administración de AWS. Para obtener una plantilla de AWS SAM que cree esta API y todos los recursos relacionados, consulte `template.yaml`.

Temas

- Paso 1: crear una tabla de DynamoDB
- Paso 2: crear una función Lambda
- Paso 3: crear una API HTTP
- Paso 4: crear rutas
- Paso 5: crear una integración
- Paso 6: conectar la integración a las rutas
- Paso 7: probar la API
- Paso 8: Eliminar
- Pasos siguientes: automatización con AWS SAM o CloudFormation

Paso 1: crear una tabla de DynamoDB

Se utiliza una tabla DynamoDB para almacenar datos para la API.

Cada elemento tiene un ID único, que usamos como clave de partición para la tabla.

Cree una tabla de DynamoDB

1. Abra la consola de DynamoDB en https://console.aws.amazon.com/dynamodb/.
2. Seleccione Create table.
3. En Nombre de la tabla, introduzca **`http-crud-tutorial-items`**.
4. En Partition key (Clave de partición), ingrese **`id`**.
5. Seleccione Create table (Creación de tabla).

Paso 2: crear una función Lambda

Se crea una función Lambda para el backend de la API. Esta función Lambda crea, lee, actualiza y elimina elementos de DynamoDB. La función utiliza eventos de API Gateway para determinar cómo interactuar con DynamoDB. Para simplificar el proceso, este tutorial utiliza una sola función Lambda. Como práctica recomendada, se deben crear funciones separadas para cada ruta. Para obtener más información sobre Lambda, consulte The Lambda monolith (El monolito de Lambda).

Cómo crear una función de Lambda

1. Inicie sesión en la consola de Lambda en https://console.aws.amazon.com/lambda/.
2. Elija Create function (Crear función).
3. En Function name (Nombre de función), introduzca **`http-crud-tutorial-function`**.
4. En Tiempo de ejecución, elija el último tiempo de ejecución de Node.js o Python compatible.
5. En Permisos, seleccione Cambiar el rol de ejecución predeterminado.
6. Seleccione Crear un nuevo rol en plantillas de políticas de AWS.
7. En Nombre del rol, introduzca **`http-crud-tutorial-role`**.
8. En Plantillas de políticas, seleccione **`Simple microservice permissions`**. Esta política concede a la función Lambda permiso para interactuar con DynamoDB.

> **Note**
>
> Para simplificar el proceso, este tutorial utiliza una política administrada. Como práctica recomendada, se deben crear políticas de IAM propias para otorgar los permisos mínimos requeridos.

9. Elija Create function (Crear función).
10. Abra la función de Lambda en el editor de código de la consola y sustituya su contenido con el siguiente código. Seleccione Implementar para actualizar la función.

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
```

Paso 3: crear una API HTTP

La API HTTP proporciona un punto de enlace HTTP para su función de Lambda. En este paso, se crea una API vacía. En los siguientes pasos, se configuran rutas e integraciones para conectar la API y la función Lambda.

Para crear una API HTTP

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Seleccione Crear APIy, a continuación, para API HTTP, seleccione Crear.
3. En API name (Nombre de la API), escriba**`http-crud-tutorial-api`**.
4. En Tipo de dirección IP, seleccione IPv4.
5. Elija Next (Siguiente).

6. En Configurar rutas, seleccione Siguiente para omitir la creación de rutas. Se crearán rutas más adelante.
7. Revise la etapa que API Gateway crea y, a continuación, seleccione Siguiente.
8. Seleccione Create (Crear).

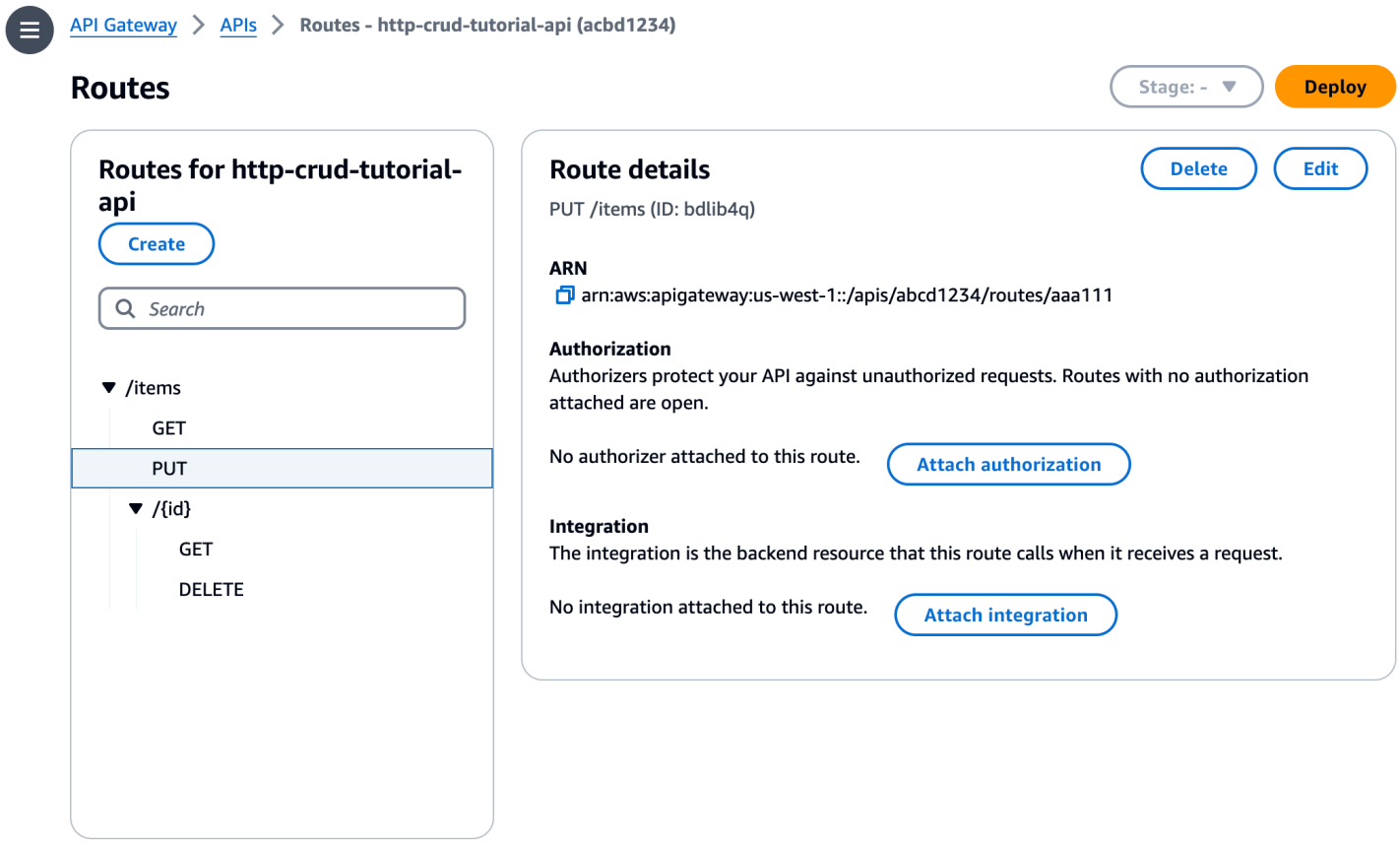
Paso 4: crear rutas

Las rutas son una manera de enviar solicitudes entrantes de API a los recursos de backend. Las rutas constan de dos partes: un método HTTP y una ruta de recurso, por ejempl, `GET /items`. Para este ejemplo de API, creamos cuatro rutas:

- `GET /items/{id}`
- `GET /items`
- `PUT /items`
- `DELETE /items/{id}`

Para crear rutas

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija la API.
3. Elija Routes (Rutas).
4. Seleccione Create (Crear).
5. En Method (Método), seleccione **`GET`**.
6. En la ruta de acceso, introduzca **`/items/{id}`**. El `{id}` al final de la ruta es un parámetro de ruta que API Gateway recupera de la ruta de solicitud cuando un cliente realiza una solicitud.
7. Seleccione Create (Crear).
8. Repita los pasos 4 a 7 para `GET /items`, `DELETE /items/{id}`, y `PUT /items`.



Paso 5: crear una integración

Se crea una integración para conectar una ruta a los recursos de backend. Para este ejemplo de API, se crea una integración Lambda que se utiliza para todas las rutas.

Para crear una integración

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija la API.
3. Seleccione Integraciones.
4. Seleccione Administrar integraciones y, a continuación, seleccione Crear.
5. Omitir Conectar esta integración a una ruta. Esta etapa se completará más adelante.
6. En Tipo de integración, seleccione Función Lambda.
7. En Función Lambda, introduzca **`http-crud-tutorial-function`**.
8. Seleccione Create (Crear).

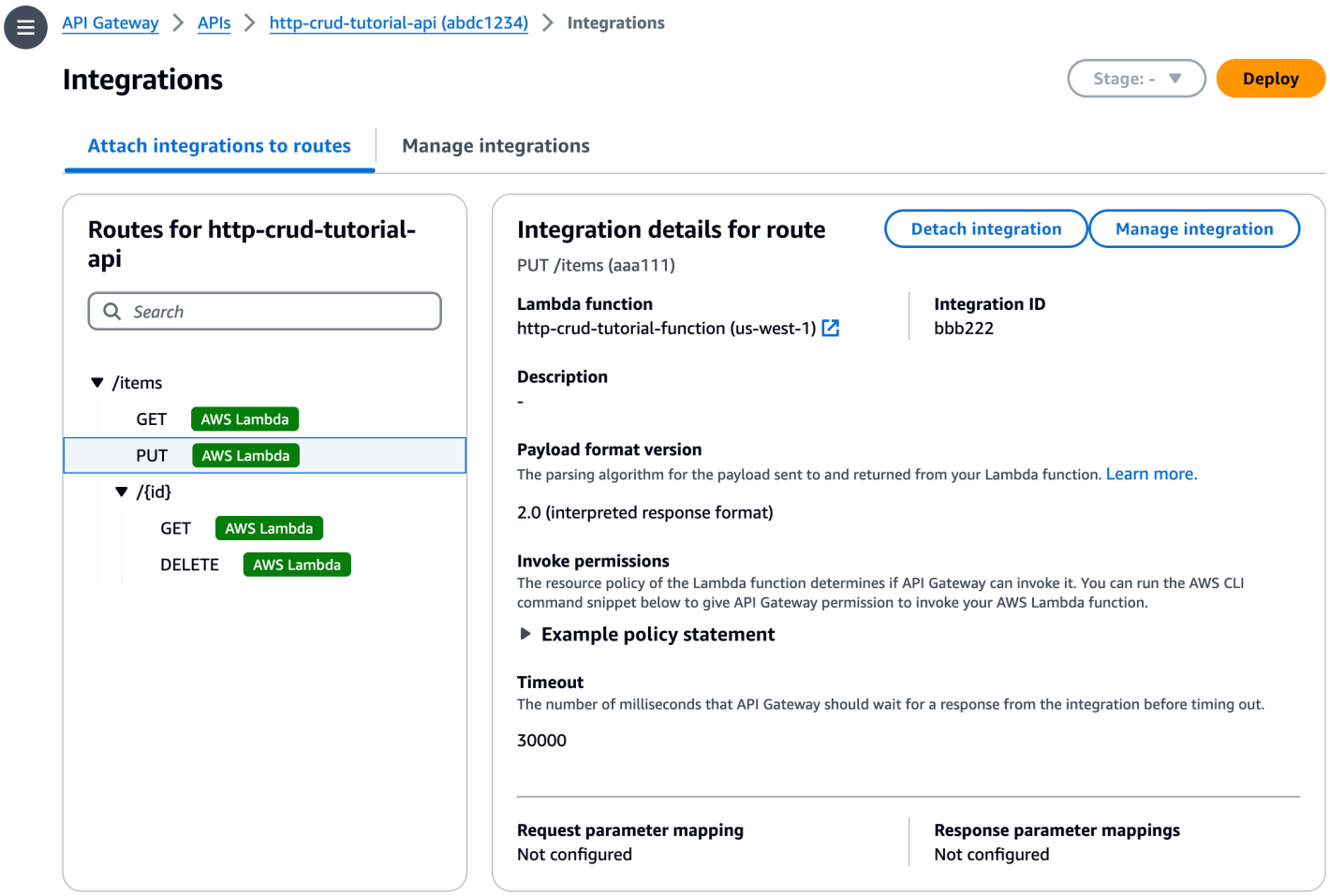
Paso 6: conectar la integración a las rutas

Para este ejemplo de API, se utiliza la misma integración Lambda para todas las rutas. Después de conectar la integración a todas las rutas de la API, la función Lambda se invoca cuando un cliente llama a cualquiera de sus rutas.

Para conectar integraciones a rutas

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija la API.
3. Seleccione Integraciones.
4. Seleccione una ruta.
5. En Elegir una integración existente, seleccione **`http-crud-tutorial-function`**.
6. Seleccione Conectar integración.
7. Repita los pasos 4 a 6 para todas las rutas.

Todas las rutas muestran que se adjuntó una integración de AWS Lambda.



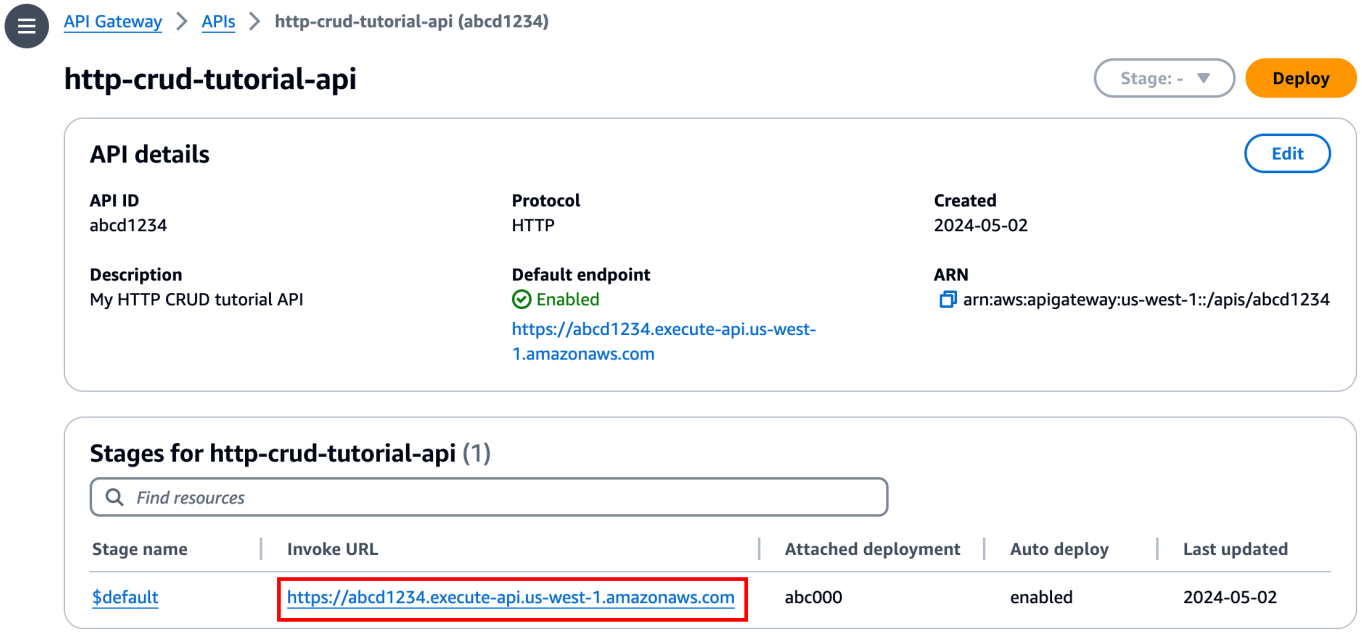
Ahora que se tiene una API HTTP con rutas e integraciones, se puede probar la API.

Paso 7: probar la API

Para asegurarse de que la API está funcionando, se utiliza curl.

Para obtener la URL para invocar la API

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija la API.
3. Tenga en cuenta la URL de invocación de la API. Aparece en Invocar URL en la página Detalles.



4. Copie la URL de invocación de la API.

   La totalidad de la URL se parece a `https://abcdef123.execute-api.us-west-2.amazonaws.com`.

Para crear o actualizar un elemento

- Utilice el siguiente comando para crear o actualizar un artículo. El comando incluye un cuerpo de solicitud con el ID, el precio y el nombre del artículo.

```
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
```

Para ver todos los artículos

- Utilice el siguiente comando para enumerar todos los artículos.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/items
```

Para ver un artículo

- Utilice el siguiente comando para ver un artículo por su ID.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/items/123
```

Para eliminar un elemento

1. Utilice el siguiente comando para eliminar un artículo.

```
curl -X "DELETE" https://abcdef123.execute-api.us-west-2.amazonaws.com/items/123
```

2. Ver todos los artículos para verificar que el artículo se eliminó.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/items
```

Paso 8: Eliminar

Para evitar costos innecesarios, elimine los recursos creados como parte de este ejercicio introductorio. Los siguientes pasos eliminan la API HTTP, la función de Lambda y los recursos asociados.

Para eliminar una tabla de DynamoDB

1. Abra la consola de DynamoDB en https://console.aws.amazon.com/dynamodb/.
2. Seleccionar la tabla.
3. Elija Delete table (Eliminar tabla).
4. Confirme la elección y seleccione Eliminar.

Para eliminar una API HTTP

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. En la página API, seleccione una API. Seleccione Actions y, luego, Delete.
3. Elija Eliminar.

Para eliminar una función de Lambda

1. Inicie sesión en la consola de Lambda en https://console.aws.amazon.com/lambda/.
2. En la página Functions (Funciones), seleccione una función. Seleccione Actions y, luego, Delete.
3. Elija Eliminar.

Para eliminar el grupo de registro de una función de Lambda

1. En la consola de Amazon CloudWatch, abra la página de grupos de registro.
2. En la página Grupos de registro, seleccione el grupo de registro de la función (`/aws/lambda/http-crud-tutorial-function`). Elija Actions (Acciones) y, a continuación, elija Delete log group (Eliminar grupo de registro).
3. Elija Eliminar.

Para eliminar el rol de ejecución de una función de Lambda

1. En la consola de AWS Identity and Access Management, abra la página Roles (Roles).
2. Seleccione el rol de la función, por ejempl, `http-crud-tutorial-role`.
3. Elija Delete role (Eliminar rol).
4. Elija Sí, eliminar.

Pasos siguientes: automatización con AWS SAM o CloudFormation

Puede automatizar la creación y la limpieza de los recursos de AWS mediante el uso de CloudFormation o AWS SAM. Para obtener un ejemplo de plantilla de AWS SAM para este tutorial, consulte `template.yaml`.

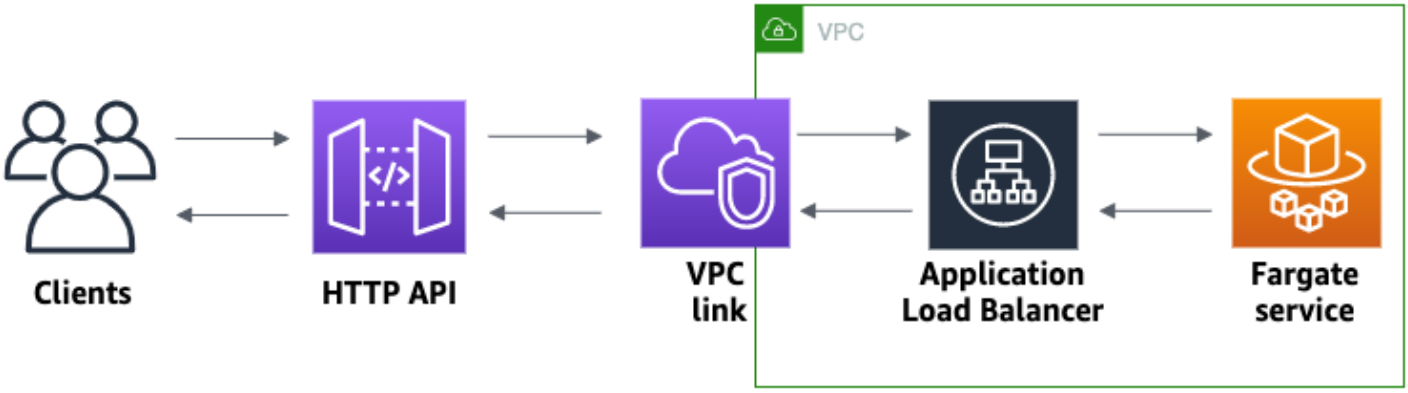
Para obtener plantillas de CloudFormation de ejemplo, consulte las plantillas de CloudFormation de ejemplo.

Tutorial: Creación de una API de HTTP con una integración privada en un servicio de Amazon ECS

En este tutorial, creará una API sin servidor que se conecta a un Servicio Amazon ECS que se ejecuta en una Amazon VPC Los clientes fuera de su VPC de Amazon pueden utilizar la API para acceder al Servicio Amazon ECS.

Para completar este tutorial se necesita aproximadamente una hora. En primer lugar, utilice una plantilla de CloudFormation para crear una nube de Amazon VPC y un servicio Amazon ECS. Luego, utilice la consola de API Gateway para crear un enlace de VPC. El vínculo VPC permite a API Gateway acceder al Servicio Amazon ECS que se ejecuta en su Amazon VPC A continuación, cree una API HTTP que utilice el vínculo VPC para conectarse al Servicio Amazon ECS. Por último, se prueba la API.

Al invocar la API HTTP, API Gateway envía la solicitud al Servicio Amazon ECS a través del vínculo VPC y, a continuación, devuelve la respuesta del servicio.



Para completar este tutorial, necesita una cuenta de AWS y un usuario de AWS Identity and Access Management con acceso a la consola. Para obtener más información, consulte Requisitos previos.

En este tutorial, se utiliza la Consola de administración de AWS. Para obtener una plantilla de CloudFormation que cree esta API y todos los recursos relacionados, consulte template.yaml.

Temas

- Paso 1: crear un Servicio Amazon ECS
- Paso 2: crear un enlace de VPC
- Paso 3: crear una API HTTP
- Paso 4: creación de una ruta
- Paso 5: crear una integración
- Paso 6: probar la API
- Paso 7: Limpieza
- Próximos pasos: automatice con CloudFormation

Paso 1: crear un Servicio Amazon ECS

Amazon ECS es un servicio de administración de contenedores que le facilita la tarea de ejecutar, detener y administrar contenedores de Docker en un clúster. En este tutorial, ejecute el clúster en una infraestructura sin servidor administrada por Amazon ECS.

Descargue y descomprima esta plantilla de CloudFormation que crea todas las dependencias del servicio, incluida una nube de Amazon VPC. Utilice la plantilla para crear un Servicio Amazon ECS que utilice un Application Load Balancer.

Para crear una pila de CloudFormation

1. Abra la consola de CloudFormation en https://console.aws.amazon.com/cloudformation.
2. Seleccione Create stack (Crear pila) y, a continuación, seleccione With new resources (standard) (Con nuevos recursos [estándar]).
3. En Specify template (Especificar plantilla), elija Upload a template file (Cargar un archivo de plantilla).
4. Seleccione la plantilla que ha descargado.
5. Elija Next (Siguiente).
6. En Stack name (Nombre de pila), escriba **`http-api-private-integrations-tutorial`** y, a continuación, elija Next (Siguiente).
7. En Configure stack options (Configurar opciones de pila), elija Next (Siguiente).
8. Para Capabilities (Capacidades), sepa que CloudFormation puede crear recursos de IAM en su cuenta.
9. Elija Siguiente y, a continuación, elija Enviar.

CloudFormation aprovisiona al Servicio ECS. Puede tardar unos minutos. Cuando el estado de la pila de CloudFormation sea CREATE_COMPLETE, estará listo para continuar con el paso siguiente.

Paso 2: crear un enlace de VPC

Un enlace de VPC permite a API Gateway acceder a recursos privados en una Amazon VPC Utilice un enlace de VPC para permitir a los clientes acceder al Servicio Amazon ECS a través de su API HTTP.

Para crear un enlace de VPC

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Enlaces de VPC y, a continuación, elija Crear.

   Es posible que necesite elegir el icono del menú para abrir el panel de navegación principal.
3. En Choose a VPC link version (Elegir una versión de enlace de VPC), seleccione VPC link for HTTP APIs (Enlace de VPC para API HTTP).
4. En Name (Nombre) escriba **`private-integrations-tutorial`**.
5. En VPC, elija la VPC que ha creado en el paso 1. El nombre debe comenzar con PrivateIntegrationsStack.
6. En Subnets (Subredes), seleccione las dos subredes privadas de la VPC. Los nombres terminan en `PrivateSubnet`.
7. Para Grupos de seguridad, seleccione el ID de grupo que comience por `private-integrations-tutorial` y tenga la descripción `PrivateIntegrationsStack/PrivateIntegrationsTutorialService/Service/SecurityGroup`.
8. Seleccione Create (Crear).

Después de crear el vínculo VPC, API Gateway aprovisiona las interfaces de red elásticas para acceder a la VPC. El proceso puede demorar unos minutos. Mientras tanto, puede crear su API.

Paso 3: crear una API HTTP

La API HTTP proporciona un punto de enlace HTTP para su Servicio Amazon ECS. En este paso, se crea una API vacía. En los pasos 4 y 5, va a configurar una ruta y una integración para conectar la API y el Servicio Amazon ECS.

Para crear una API HTTP

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Seleccione Crear APIy, a continuación, para API HTTP, seleccione Crear.
3. En API name (Nombre de la API), escriba**`http-private-integrations-tutorial`**.
4. En Tipo de dirección IP, seleccione IPv4.
5. Elija Next (Siguiente).

6. En Configurar rutas, seleccione Siguiente para omitir la creación de rutas. Se crearán rutas más adelante.
7. Revise la etapa que API Gateway le crea. API Gateway crea una etapa `$default` con implementaciones automáticas habilitadas, que es la mejor opción para este tutorial. Elija Next (Siguiente).
8. Seleccione Create (Crear).

Paso 4: creación de una ruta

Las rutas son una manera de enviar solicitudes entrantes de API a los recursos de backend. Las rutas constan de dos partes: un método HTTP y una ruta de recurso, por ejempl, `GET /items`. Para este ejemplo de API, creamos una ruta.

Para crear una ruta

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija la API.
3. Elija Routes (Rutas).
4. Seleccione Create (Crear).
5. En Method (Método), seleccione **`ANY`**.
6. En la ruta de acceso, introduzca **`/{proxy+}`**. El `{proxy+}` al final de la ruta es una variable de ruta voraz. API Gateway envía todas las solicitudes a su API por esta ruta.
7. Seleccione Create (Crear).

Paso 5: crear una integración

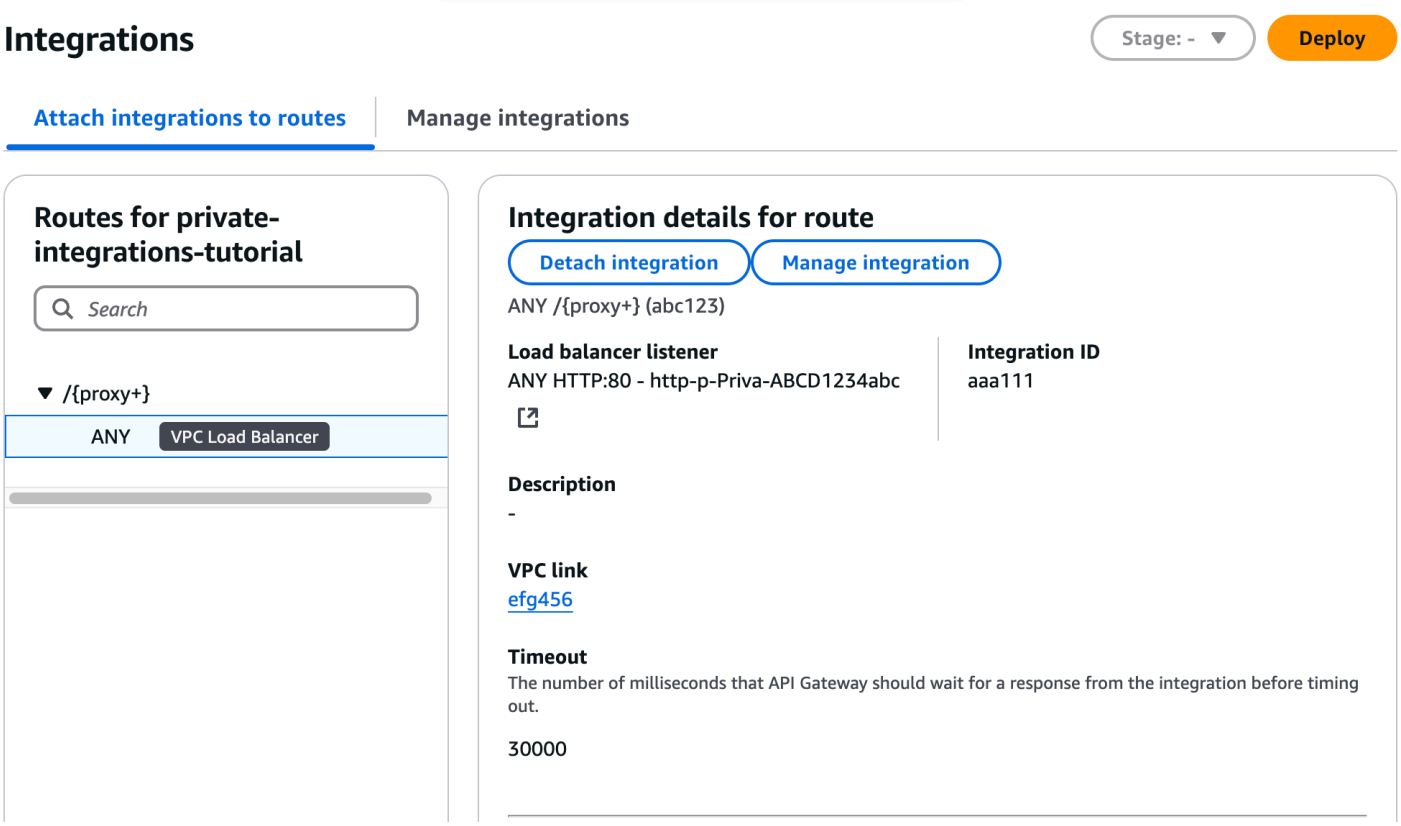
Se crea una integración para conectar una ruta a los recursos de backend.

Para crear una integración

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija la API.
3. Seleccione Integraciones.
4. Seleccione Administrar integraciones y, a continuación, seleccione Crear.

5. En Atach this intergration to a route (Adjuntar esta integración a una ruta), seleccione la ruta ANY/{proxy+} que creó anteriormente.
6. En Integration type (Tipo de integración), elija Private resource (Recurso privado).
7. En Integration details (Detalles de integración), elija Select manually (Seleccionar manualmente).
8. En Target service (Servicio de destino), seleccione ALB/NLB.
9. En Load balancer (Equilibrador de carga), elija el equilibrador de carga que creó con la plantilla de CloudFormation en el paso 1. El nombre debería comenzar con http-Priva.
10. En Listener (Agente de escucha), elija **`HTTP 80`**.
11. Para VPC link (Enlace de VPC), elija el enlace de VPC que creó en el paso 2. El nombre debería ser `private-integrations-tutorial`.
12. Seleccione Create (Crear).

Para comprobar que la ruta y la integración están configuradas correctamente, seleccione Attach integrations to routes (Adjuntar las integraciones a las rutas). La consola muestra que tiene una ruta `ANY /{proxy+}` con una integración a un balanceador de carga de VPC.



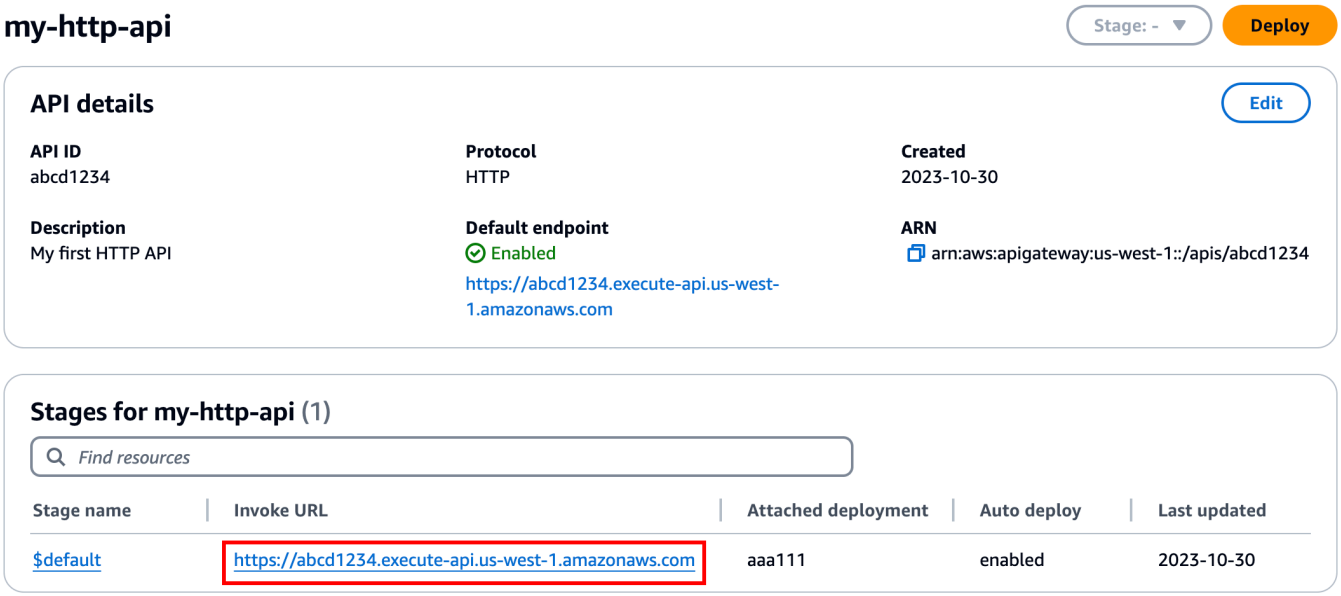
Ya está listo para probar su API.

Paso 6: probar la API

A continuación, pruebe su API para asegurarse de que se encuentra en funcionamiento. Para mayor simplicidad, utilice un navegador web para invocar la API.

Para probar la API

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija la API.
3. Tenga en cuenta la URL de invocación de la API.



4. En un navegador web, vaya a la URL que invoca a su API.

   La URL completa debería ser `https://abcdef123.execute-api.us-east-2.amazonaws.com`.

   Su navegador envía una `GET` solicitud a la API.
5. Compruebe que la respuesta de la API sea un mensaje de bienvenida que le indique que su aplicación se está ejecutando en Amazon ECS.

   Si ve el mensaje de bienvenida, ha creado correctamente un Servicio Amazon ECS que se ejecuta en la VPC de Amazon y ha utilizado una API HTTP de API Gateway con un vínculo VPC para acceder al servicio Amazon ECS.

Paso 7: Limpieza

Para evitar costos innecesarios, elimine los recursos creados como parte de este tutorial Los pasos siguientes eliminan el enlace de la VPC, la pila de CloudFormation y la API HTTP.

Para eliminar una API HTTP

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. En la página API, seleccione una API. Elija Action (Acciones), elija Delete (Eliminar)y, a continuación, confirme su elección.

Para eliminar un enlace de VPC

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija VPC link (Enlace de VPC).
3. Seleccione el enlace de VPC, elija Delete (Eliminar)y, a continuación, confirme su elección.

Para eliminar una pila de CloudFormation

1. Abra la consola de CloudFormation en https://console.aws.amazon.com/cloudformation.
2. Seleccione su pila de CloudFormation.
3. Elija Delete (Eliminar) y, a continuación, confirme su elección.

Próximos pasos: automatice con CloudFormation

Puede automatizar la creación y la limpieza de todos los recursos de AWS involucrados en este tutorial. Para obtener una plantilla de CloudFormation de ejemplo completa, consulte template.yaml.

Tutoriales de la API REST de Amazon API Gateway

Los siguientes tutoriales proporcionan un ejercicio práctico que le ayudará a obtener información sobre las API de WebSocket de API Gateway.

Temas

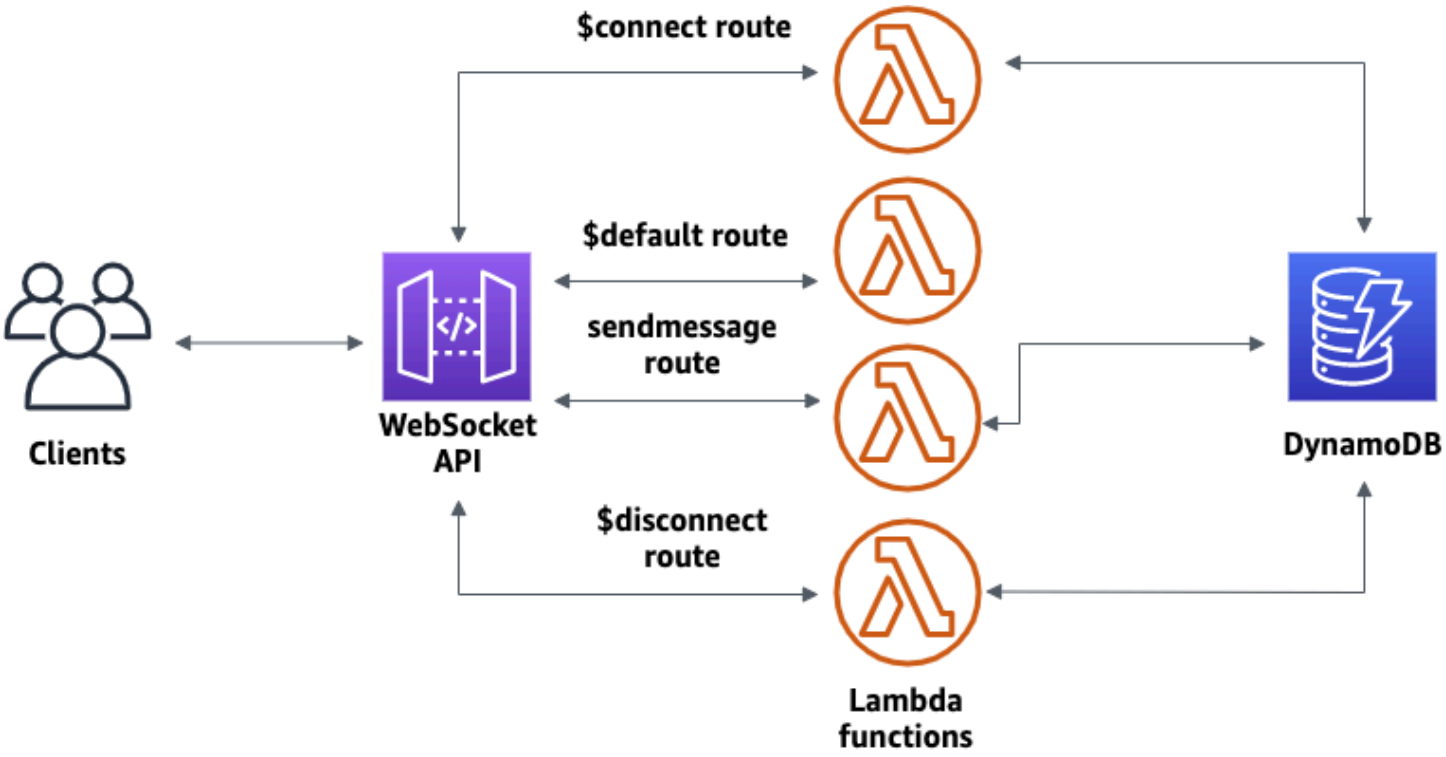
- Tutorial: Creación de una aplicación de chat de WebSocket con una API de WebSocket, Lambda y DynamoDB

- Tutorial: Creación de una API de WebSocket con una integración de AWS

Tutorial: Creación de una aplicación de chat de WebSocket con una API de WebSocket, Lambda y DynamoDB

En este tutorial, creará una aplicación de chat sin servidor con una API de WebSocket. Con una API de WebSocket, puede admitir la comunicación bidireccional entre clientes. Los clientes pueden recibir mensajes sin tener que sondear para comprobar si hay actualizaciones.

Se tardan aproximadamente 30 minutos en completar este tutorial. En primer lugar, usará una plantilla de CloudFormation para crear funciones Lambda que gestionarán las solicitudes de API, así como una tabla de DynamoDB que almacena los ID de cliente. A continuación, utilizará la consola de API Gateway para crear una API de WebSocket que se integra con las funciones Lambda. Por último, probarás la API para verificar que se envían y reciben mensajes.



Para completar este tutorial, necesita una cuenta de AWS y un usuario de AWS Identity and Access Management con acceso a la consola. Para obtener más información, consulte Requisitos previos.

También es necesario `wscat` para conectarse a la API. Para obtener más información, consulte the section called “Utilice `wscat` para conectarse y enviar mensajes a una API de WebSocket”.

Temas

- Paso 1: Crear funciones Lambda y una tabla de DynamoDB
- Paso 2: Crear una API de WebSocket
- Paso 3: probar la API
- Paso 4: Limpiar
- Próximos pasos: automatice con CloudFormation

Paso 1: Crear funciones Lambda y una tabla de DynamoDB

Descargue y descomprima la plantilla de creación de aplicaciones para CloudFormation. Utilizará esta plantilla para crear una tabla de Amazon DynamoDB para almacenar los ID de cliente de la aplicación. Cada cliente conectado tiene un ID único que utilizaremos como clave de partición de la tabla. Esta plantilla también crea funciones Lambda que actualizan las conexiones de clientes en DynamoDB y gestionan el envío de mensajes a los clientes conectados.

Para crear una pila de CloudFormation

1. Abra la consola de CloudFormation en https://console.aws.amazon.com/cloudformation.
2. Seleccione Create stack (Crear pila) y, a continuación, seleccione With new resources (standard) (Con nuevos recursos [estándar]).
3. En Specify template (Especificar plantilla), elija Upload a template file (Cargar un archivo de plantilla).
4. Seleccione la plantilla que ha descargado.
5. Elija Next (Siguiente).
6. En Stack name (Nombre de pila), escriba **`websocket-api-chat-app-tutorial`** y, a continuación, elija Next (Siguiente).
7. En Configure stack options (Configurar opciones de pila), elija Next (Siguiente).
8. Para Capabilities (Capacidades), sepa que CloudFormation puede crear recursos de IAM en su cuenta.
9. Elija Siguiente y, a continuación, elija Enviar.

CloudFormation aprovisiona los recursos especificados en la plantilla. Puede tardar varios minutos en finalizar el aprovisionamiento de los recursos. Cuando el estado de la pila de CloudFormation sea CREATE_COMPLETE, estará listo para continuar con el paso siguiente.

Paso 2: Crear una API de WebSocket

Creará una API de WebSocket para gestionar las conexiones de clientes y enrutar solicitudes a las funciones Lambda que creó en el paso 1.

Para crear una API de WebSocket

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Seleccione Crear API. En WebSocket API (API de WebSocket), elija Build (Crear).
3. En API name (Nombre de la API), escriba**`websocket-chat-app-tutorial`**.
4. En Tipo de dirección IP, seleccione IPv4.
5. Para Route selection expression (Expresión de selección de ruta), ingrese **`request.body.action`**. La expresión de selección de ruta determina la ruta que API Gateway invoca cuando un cliente envía un mensaje.
6. Elija Siguiente.
7. Para Predefined routes (Rutas predefinidas), elija Add $connect (Agregar $connect), Add $disconnect (Agregar $disconnect) y Add $default (Agregar $default). Las rutas $connect y $disconnect son rutas especiales que API Gateway invoca automáticamente cuando un cliente se conecta o se desconecta de una API. API Gateway invoca la ruta `$default` cuando ninguna otra ruta coincide con una solicitud.
8. Para Custom routes (Rutas personalizadas), elija Add custom route (Agregar ruta personalizada). Para Route key (Clave de ruta), ingrese **`sendmessage`**. Esta ruta personalizada gestiona los mensajes que se envían a los clientes conectados.
9. Elija Siguiente.
10. En Attach integrations (Adjuntar integraciones), para cada ruta e Integration type (Tipo de integración), elija Lambda.

    Para Lambda, elija la función Lambda correspondiente que creó con CloudFormation en el paso 1. El nombre de cada función coincide con una ruta. Por ejemplo, para la ruta $connect, elija la función denominada **`websocket-chat-app-tutorial-ConnectHandler`**.
11. Revise la etapa que API Gateway le crea. De forma predeterminada, API Gateway crea un nombre de etapa `production` e implementa automáticamente la API en esa fase. Elija Next (Siguiente).
12. Elija Create and deploy (Crear e implementar).

Paso 3: probar la API

A continuación, pruebe la API para asegurarse de que funciona correctamente. Utilice el comando `wscat` para conectarse a la API.

Para obtener la URL para invocar la API

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija la API.
3. Elija Stages (Etapas) y, a continuación, elija production (producción).
4. Tenga en cuenta la WebSocket URL (URL de WebSocket) de la API. La dirección URL debe tener un aspecto similar al siguiente: `wss://abcdef123.execute-api.us-east-2.amazonaws.com/production`.

Para conectarse a la API

1. Para conectarse a la API, utilice el siguiente comando. Cuando se conecte a la API, API Gateway invoca la ruta `$connect`. Cuando se invoca esta ruta, llama a una función Lambda que almacena el ID de conexión en DynamoDB.

   ```
   wscat -c wss://abcdef123.execute-api.us-west-2.amazonaws.com/production
   ```

   ```
   Connected (press CTRL+C to quit)
   ```
2. Abra un nuevo terminal y ejecute el comando wscat de nuevo con los siguientes parámetros.

   ```
   wscat -c wss://abcdef123.execute-api.us-west-2.amazonaws.com/production
   ```

   ```
   Connected (press CTRL+C to quit)
   ```

   Esto le proporciona dos clientes conectados que pueden intercambiar mensajes.

Cómo enviar un mensaje

- API Gateway determina qué ruta invocar en función de la expresión de selección de rutas de la API. La expresión de selección de rutas de la API es `$request.body.action`. Como resultado, API Gateway invoca la ruta `sendmessage` cuando envía el siguiente mensaje:

```
{"action": "sendmessage", "message": "hello, everyone!"}
```

La función Lambda asociada a la ruta invocada recopila los ID de cliente de DynamoDB. A continuación, la función llama a la API de administración de API Gateway y envía el mensaje a esos clientes. Todos los clientes conectados reciben el siguiente mensaje:

```
< hello, everyone!
```

Para invocar la ruta $default de la API

- API Gateway invoca la ruta predeterminada de la API cuando un cliente envía un mensaje que no coincide con las rutas definidas. La función Lambda asociada a la ruta `$default` utiliza la API de administración de API Gateway para enviar información al cliente sobre la conexión.

```
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
```

Para desconectarse de la API

- Pulse **CTRL+C** para desconectarse de la API. Cuando un cliente se desconecta de la API, API Gateway invoca la ruta `$disconnect` de la API. La integración de Lambda para la ruta `$disconnect` de la API elimina el ID de conexión de DynamoDB.

Paso 4: Limpiar

Para evitar costos innecesarios, elimine los recursos creados como parte de este tutorial Los siguientes pasos eliminan la pila y API de WebSocket de CloudFormation.

Para eliminar una API de WebSocket

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.

2. En la página de las API, seleccione la API `websocket-chat-app-tutorial`. Elija Action (Acciones), elija Delete (Eliminar)y, a continuación, confirme su elección.

Para eliminar una pila de CloudFormation

1. Abra la consola de CloudFormation en https://console.aws.amazon.com/cloudformation.
2. Seleccione su pila de CloudFormation.
3. Elija Delete (Eliminar) y, a continuación, confirme su elección.

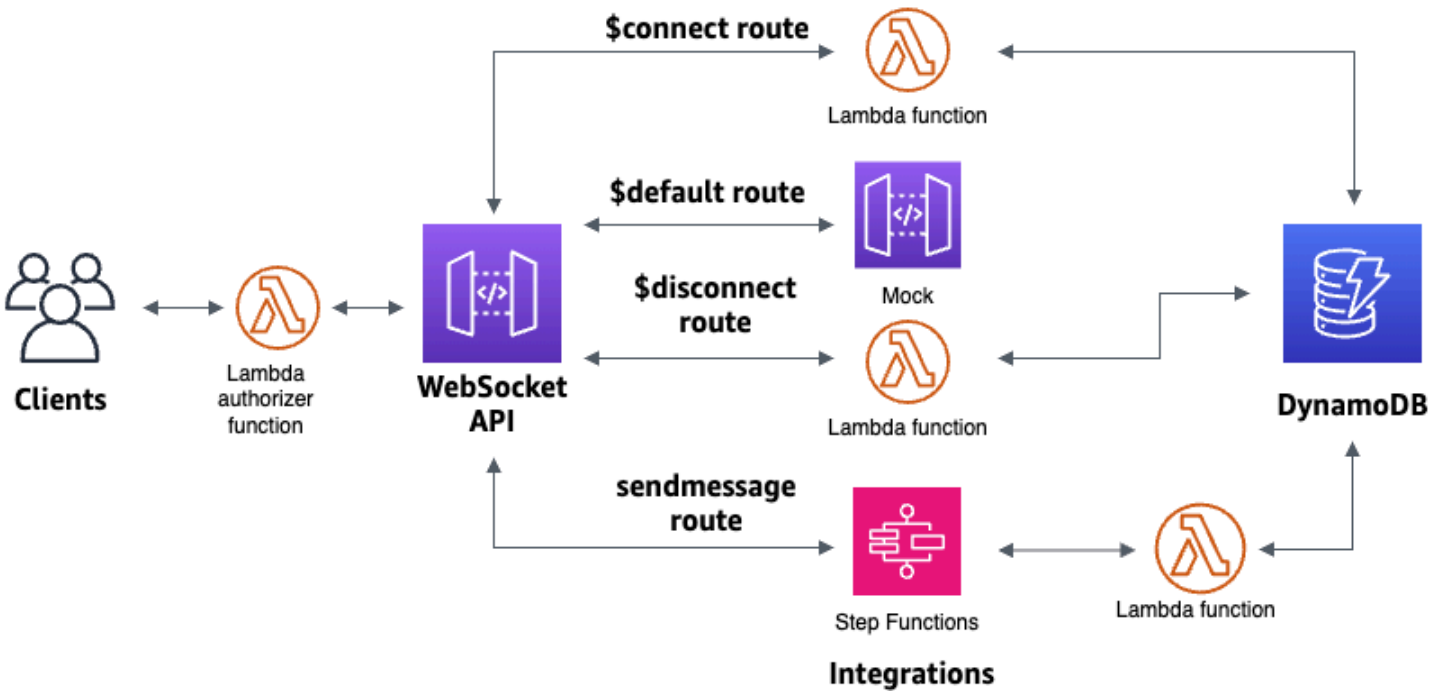
Próximos pasos: automatice con CloudFormation

Puede automatizar la creación y la limpieza de todos los recursos de AWS involucrados en este tutorial. Para obtener una plantilla de CloudFormation que cree esta API y todos los recursos relacionados, consulte ws-chat-app.yaml.

Tutorial: Creación de una API de WebSocket con una integración de AWS

En este tutorial, se crea una aplicación de transmisión sin servidor con una API de WebSocket. Los clientes pueden recibir mensajes sin tener que sondear para comprobar si hay actualizaciones.

Este tutorial muestra cómo transmitir mensajes a los clientes conectados e incluye un ejemplo de un autorizador de Lambda, una integración simulada y una integración que no sea de proxy en Step Functions.



Una vez haya creado los recursos con una plantilla de CloudFormation, utilizará la consola de API Gateway para crear una API de WebSocket que se integre con los recursos de AWS. Adjuntará un autorizador de Lambda a la API y creará una integración de servicios de AWS con Step Functions para iniciar la ejecución de una máquina de estado. La máquina de estado de Step Functions invocará una función de Lambda que envía un mensaje a todos los clientes conectados.

Después de crear la API, probará la conexión a la API y verificará que se envían y reciben mensajes. Para completar este tutorial se necesita aproximadamente 45 minutos.

Temas

- Requisitos previos
- Paso 1: Crear recursos
- Paso 2: Crear una API de WebSocket
- Paso 3: Creación de un autorizador de Lambda
- Paso 4: Creación de una integración bidireccional simulada
- Paso 5: Creación de una integración que no sea de proxy con Step Functions
- Paso 6: probar la API
- Paso 7: limpiar
- Siguientes pasos

Requisitos previos

Necesita los siguientes requisitos previos:

- Una cuenta de AWS y un usuario de AWS Identity and Access Management con acceso a la consola. Para obtener más información, consulte Requisitos previos.
- `wscat` para conectarse a la API. Para obtener más información, consulte the section called “Utilice `wscat` para conectarse y enviar mensajes a una API de WebSocket”.

Le recomendamos que complete el tutorial de la aplicación de chat de WebSocket antes de iniciar este tutorial. Para completar el tutorial de la aplicación de chat de WebSocket, consulte the section called “Tutorial: Creación de una aplicación de chat de WebSocket”.

Paso 1: Crear recursos

Descargue y descomprima la plantilla de creación de aplicaciones para CloudFormation. Usará esta plantilla para crear lo siguiente:

- Funciones de Lambda que gestionan las solicitudes de la API y autorizan el acceso a la API.
- Una tabla de DynamoDB para almacenar los ID de cliente y la identificación de usuario de entidad principal devuelta por el autorizador de Lambda.
- Una máquina de estado de Step Functions para enviar mensajes a los clientes conectados.

Para crear una pila de CloudFormation

1. Abra la consola de CloudFormation en https://console.aws.amazon.com/cloudformation.
2. Seleccione Create stack (Crear pila) y, a continuación, seleccione With new resources (standard) (Con nuevos recursos [estándar]).
3. En Specify template (Especificar plantilla), elija Upload a template file (Cargar un archivo de plantilla).
4. Seleccione la plantilla que ha descargado.
5. Elija Next (Siguiente).
6. En Stack name (Nombre de pila), escriba **`websocket-step-functions-tutorial`** y, a continuación, elija Next (Siguiente).
7. En Configure stack options (Configurar opciones de pila), elija Next (Siguiente).

8. Para Capabilities (Capacidades), sepa que CloudFormation puede crear recursos de IAM en su cuenta.
9. Elija Siguiente y, a continuación, elija Enviar.

CloudFormation aprovisiona los recursos especificados en la plantilla. Puede tardar varios minutos en finalizar el aprovisionamiento de los recursos. Elija la pestaña Salidas para ver los recursos creados y los ARN. Cuando el estado de la pila de CloudFormation sea CREATE_COMPLETE, estará listo para continuar con el paso siguiente.

Paso 2: Crear una API de WebSocket

Creará una API de WebSocket para gestionar las conexiones de clientes y enrutar solicitudes a los recursos que creó en el paso 1.

Para crear una API de WebSocket

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Seleccione Crear API. En WebSocket API (API de WebSocket), elija Build (Crear).
3. En API name (Nombre de la API), escriba**`websocket-step-functions-tutorial`**.
4. En Tipo de dirección IP, seleccione IPv4.
5. Para Route selection expression (Expresión de selección de ruta), ingrese **`request.body.action`**.

   La expresión de selección de ruta determina la ruta que API Gateway invoca cuando un cliente envía un mensaje.
6. Elija Siguiente.
7. Para Rutas predefinidas, elija Agregar $connect, Agregar $disconnect y Agregar $default.

   Las rutas $connect y $disconnect son rutas especiales que API Gateway invoca automáticamente cuando un cliente se conecta o se desconecta de una API. API Gateway invoca la ruta $default cuando ninguna otra ruta coincide con una solicitud. Creará una ruta personalizada para conectarse a Step Functions después de crear la API.
8. Elija Siguiente.
9. Para Integración para $connect, haga lo siguiente:

   a. Para Tipo de integración, elija Lambda.

b. Para Función de Lambda, elija la función de Lambda $connect correspondiente que creó con CloudFormation en el paso 1. El nombre de la función de Lambda debe comenzar con **`websocket-step`**.

10. Para Integración para $disconnect, haga lo siguiente:

    a. Para Tipo de integración, elija Lambda.

    b. Para Función de Lambda, elija la función de Lambda $disconnect correspondiente que creó con CloudFormation en el paso 1. El nombre de la función de Lambda debe comenzar con **`websocket-step`**.

11. Para Integración para $default, elija simulación.

    En una integración simulada, API Gateway administra la respuesta de la ruta sin un backend de integración.

12. Elija Siguiente.
13. Revise la etapa que API Gateway le crea. De forma predeterminada, API Gateway crea una etapa denominada producción e implementa automáticamente la API en esa etapa. Elija Siguiente.
14. Elija Create and deploy (Crear e implementar).

Paso 3: Creación de un autorizador de Lambda

Para controlar el acceso a la API de WebSocket, debe crear un autorizador de Lambda. La plantilla de CloudFormation creó la función del autorizador de Lambda para usted. Puede ver la función de Lambda en la consola de Lambda. El nombre no debe comenzar por **`websocket-step-functions-tutorial-AuthorizerHandler`**. Esta función de Lambda deniega todas las llamadas a la API de WebSocket a menos que el encabezado de `Authorization` sea `Allow`. La función de Lambda también transfiere la variable `$context.authorizer.principalId` a la API, que luego se utiliza en la tabla de DynamoDB para identificar a los intermediarios de la API.

En este paso, debe configurar la ruta $connect para usar el autorizador de Lambda.

Creación de un autorizador de Lambda

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Autorizadores.
3. Elija Creación de un autorizador.

4. Para Nombre del autorizador, ingrese **`LambdaAuthorizer`**.
5. Para ARN de autorizador, ingrese el nombre del autorizador creado por la plantilla de CloudFormation. El nombre no debe comenzar por **`websocket-step-functions-tutorial-AuthorizerHandler`**.

> **Note**
>
> Le recomendamos que no utilice este autorizador de ejemplo para las API de producción.

6. Para Tipo de origen de identidad, elija Encabezado. En Clave, escriba **`Authorization`**.
7. Elija Crear autorizador.

Después de crear el autorizador, debe adjuntarlo a la ruta $connect de la API.

Asociación de un autorizador a la ruta de $connect

1. En el panel de navegación principal, elija Rutas.
2. Elija la ruta $connect.
3. En la sección Configuración de la solicitud de ruta, elija Editar.
4. Para Autorización, elija el menú desplegable y, a continuación, seleccione el autorizador de solicitudes.
5. Seleccione Save changes (Guardar cambios).

Paso 4: Creación de una integración bidireccional simulada

A continuación, debe crear la integración simulada bidireccional para la ruta $default. Una integración simulada le permite enviar una respuesta al cliente sin usar un backend. Cuando crea una integración para la ruta $default, puede mostrar a los clientes cómo interactuar con la API.

Debe configurar la ruta $default para indicar a los clientes que usen la ruta sendmessage.

Creación de una integración simulada

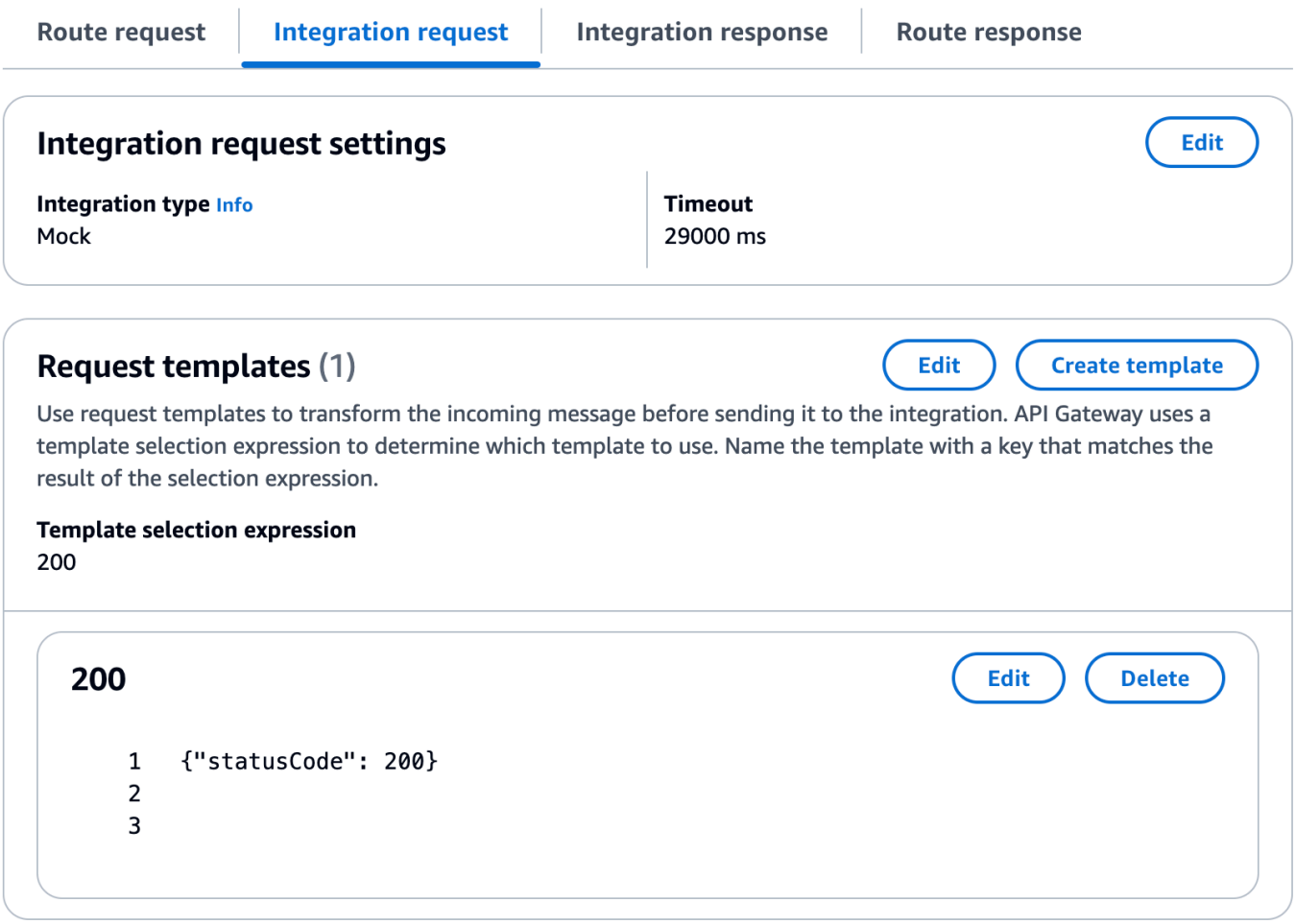
1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Elija la ruta $default y, a continuación, elija la pestaña Solicitud de integración.

3. Para las Plantillas de solicitud, elija Editar.
4. Para Expresión de selección de plantilla, ingrese **`200`** y, a continuación, elija Editar.
5. En la pestaña Solicitud de integración, para Plantillas de solicitud, elija Crear plantilla.
6. Para Clave de plantilla, ingrese **`200`**.
7. Para Generar plantilla, ingrese la siguiente plantilla de asignación:

```
{"statusCode": 200}
```

Seleccione Crear plantilla.

El resultado debe ser similar a lo siguiente:



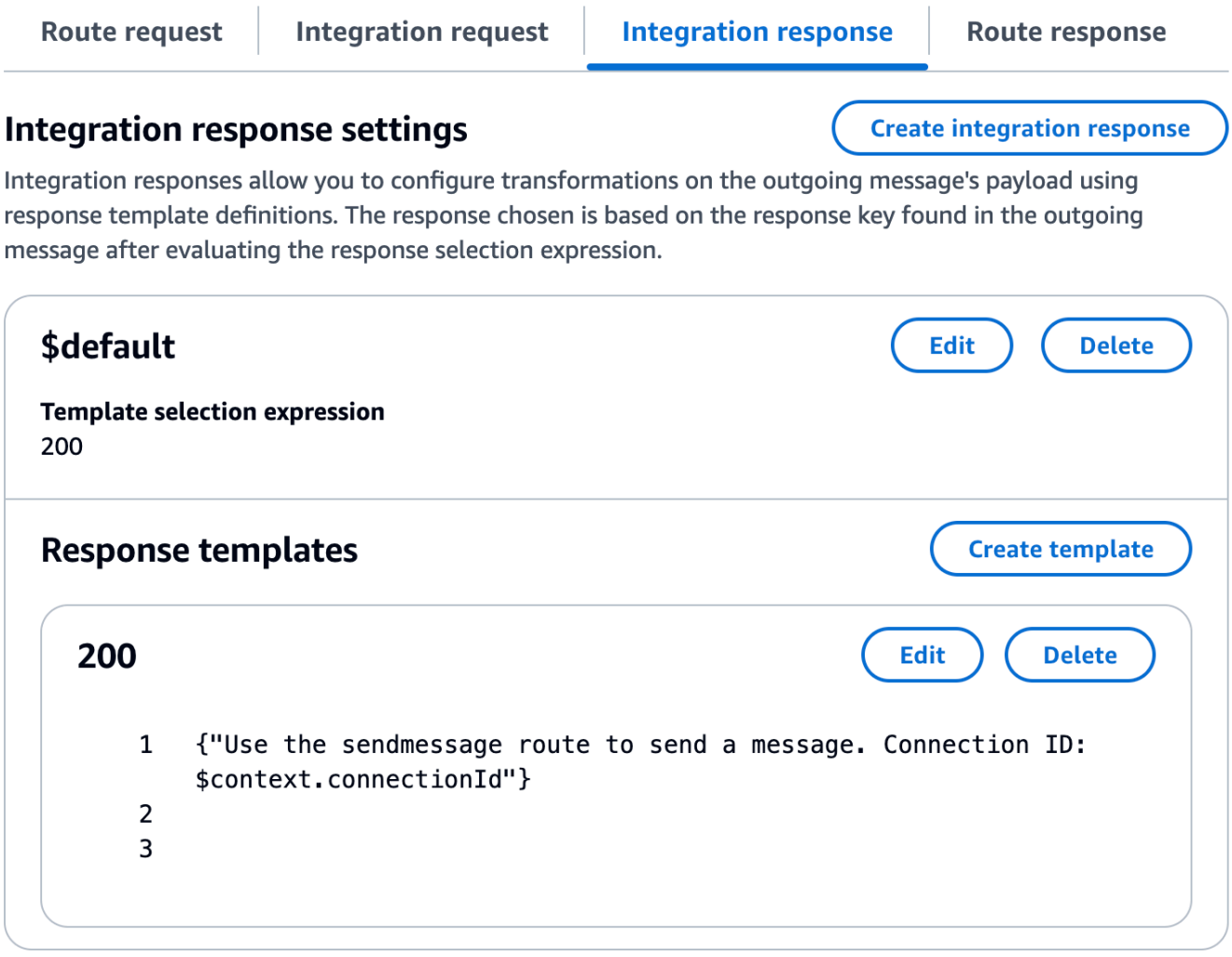
8. Para el panel de la ruta $default, elija Habilitar la comunicación bidireccional.
9. Elija la pestaña Respuesta de integración y, a continuación, elija Creación de respuesta de integración.

10. Para Clave de respuesta, escriba **`$default`**.
11. Para Expresión de selección de plantillas, ingrese **`200`**.
12. Elija Crear respuesta.
13. En Plantillas de respuesta, elija Crear plantilla.
14. Para Clave de plantilla, ingrese **`200`**.
15. Para Plantilla de respuesta, ingrese la siguiente plantilla de asignación:

```
{"Use the sendmessage route to send a message. Connection ID:
 $context.connectionId"}
```

16. Seleccione Crear plantilla.

    El resultado debe ser similar a lo siguiente:



Paso 5: Creación de una integración que no sea de proxy con Step Functions

A continuación, debe crear una ruta sendmessage. Los clientes pueden invocar la ruta sendmessage para transmitir un mensaje a todos los clientes conectados. La ruta sendmessage tiene una integración de servicio de AWS que no es de proxy con AWS Step Functions. La integración invoca el comando StartExecution para la máquina de estado de Step Functions que la plantilla de CloudFormation creó para usted.

Creación de una integración que no sea de proxy

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Elija Create route (Crear ruta).
3. Para Route key (Clave de ruta), ingrese **`sendmessage`**.
4. En Tipo de integración, seleccione Servicio de AWS.
5. Para Región de AWS, ingrese la región en la que implementó la plantilla de CloudFormation.
6. Para Servicio de AWS, elija Step Functions.
7. En Método HTTP, elija POST.
8. En Nombre de la función, introduzca **`StartExecution`**.
9. Para Rol de ejecución, ingrese el rol de ejecución creado por la plantilla de CloudFormation. El nombre debe ser WebsocketTutorialApiRole.
10. Elija Create route (Crear ruta).

A continuación, debe crear una plantilla de asignación para enviar los parámetros de la solicitud a la máquina de estado de Step Functions.

Para crear una plantilla de mapeo

1. Elija la ruta sendmessage y, a continuación, elija la pestaña Solicitud de integración.
2. En la sección Plantillas de solicitud, elija Editar.
3. Para Expresión de selección de plantillas, ingrese **`\$default`**.
4. Elija Edit (Edición de).
5. En la sección Plantillas de solicitud, elija Crear plantilla.
6. Para Clave de plantilla, ingrese **`\$default`**.
7. Para Generar plantilla, ingrese la siguiente plantilla de asignación:

```
#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"
}
```

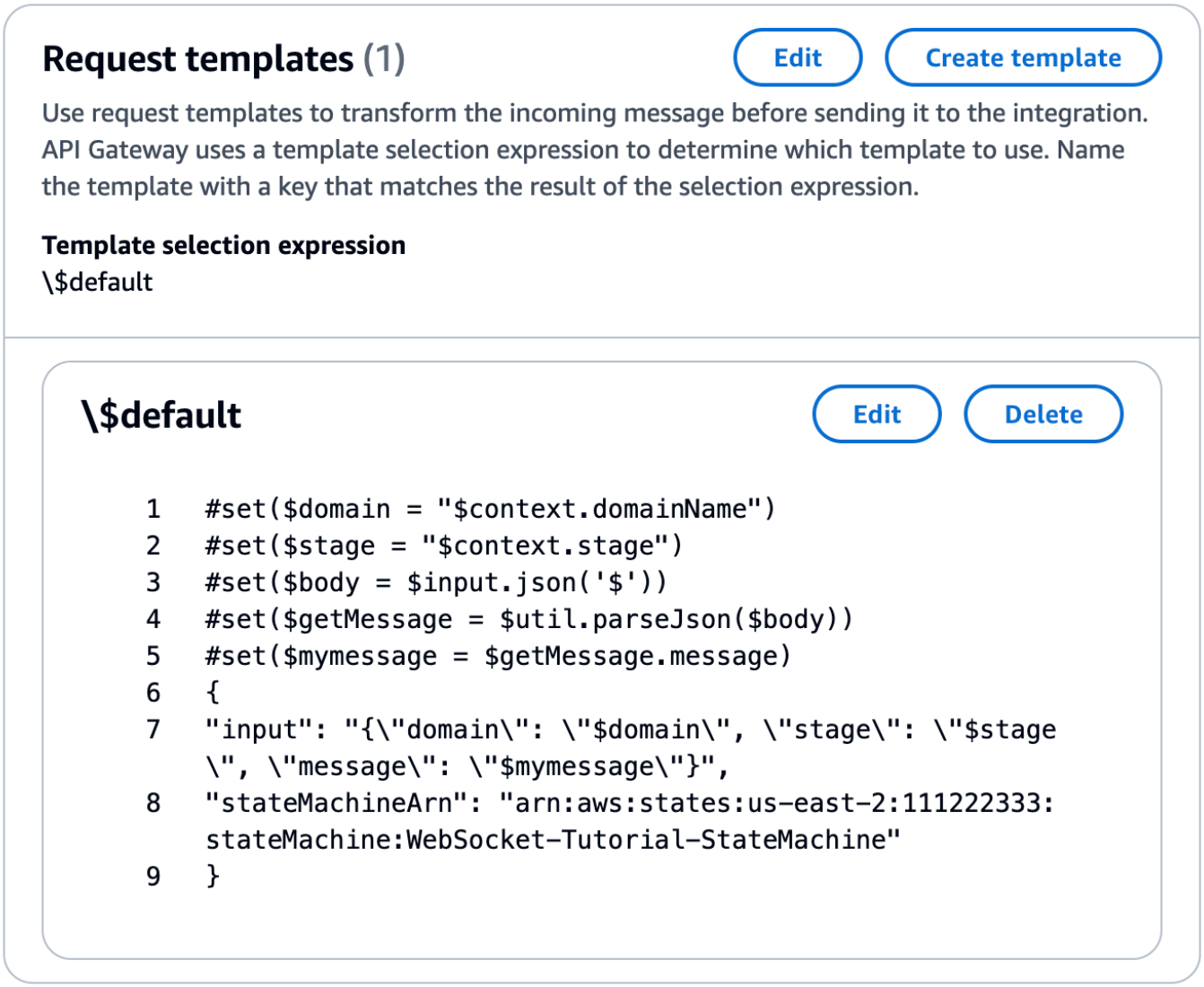
Sustituya *stateMachineArn* por el ARN de la máquina de estado creada por CloudFormation.

La plantilla de asignación hace lo siguiente:

- Crea la variable `$domain` mediante la variable de contexto `domainName`.
- Crea la variable `$stage` mediante la variable de contexto `stage`.

  Las variables `$domain` y `$stage` son necesarias para crear una URL de devolución de llamada.
- Toma el mensaje JSON `sendmessage` entrante y extrae la propiedad de `message`.
- Crea la entrada de la máquina de estado. La entrada es el dominio y la etapa de la API de WebSocket y el mensaje de la ruta `sendmessage`.

8. Seleccione Crear plantilla.



Puede crear una integración que no sea de proxy en las rutas $connect o $disconnect, para agregar o eliminar directamente un ID de conexión de la tabla de DynamoDB, sin invocar una función de Lambda.

Paso 6: probar la API

A continuación, implementará y probará la API para asegurarse de que funciona correctamente. Utilizará el comando `wscat` para conectarse a la API y, a continuación, utilizará un comando de barra inclinada para enviar un marco de ping para comprobar la conexión a la API de WebSocket.

Para implementar su API

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Rutas.
3. Elija Implementar API.
4. Para Etapa, elija producción.
5. (Opcional) Para Descripción de implementación, ingrese una descripción.
6. Elija Implementar.

Después de implementar la API, puede invocarla. Use la URL de invocación para llamar a la API.

Obtención de la URL de invocación para la API

1. Elija la API.
2. Elija Stages (Etapas) y, a continuación, elija production (producción).
3. Tenga en cuenta la WebSocket URL (URL de WebSocket) de la API. La dirección URL debe tener un aspecto similar al siguiente: `wss://abcdef123.execute-api.us-east-2.amazonaws.com/production`.

Ahora que tiene la URL de invocación, puede probar la conexión a la API de WebSocket.

Prueba de la conexión a la API

1. Para conectarse a la API, utilice el siguiente comando. En primer lugar, se prueba la conexión invocando la ruta de `/ping`.

```
wscat -c wss://abcdef123.execute-api.us-east-2.amazonaws.com/production -H
 "Authorization: Allow" --slash -P
```

```
Connected (press CTRL+C to quit)
```

2. Ingrese el siguiente comando para hacer ping al marco de control. Puede utilizar un marco de control para keepalive los fines del cliente.

```
/ping
```

El resultado debe ser similar a lo siguiente:

```
< Received pong (data: "")
```

Ahora que ha probado la conexión, se puede probar que la API funciona correctamente. En este paso, se abre una nueva ventana de terminal para que la API de WebSocket pueda enviar un mensaje a todos los clientes conectados.

Para probar la API

1. Abra un nuevo terminal y ejecute el comando `wscat` de nuevo con los siguientes parámetros.

```
wscat -c wss://abcdef123.execute-api.us-east-2.amazonaws.com/production -H
 "Authorization: Allow"
```

```
Connected (press CTRL+C to quit)
```

2. API Gateway determina qué ruta invocar en función de la expresión de selección de solicitud de rutas de la API. La expresión de selección de rutas de la API es `$request.body.action`. Como resultado, API Gateway invoca la ruta `sendmessage` cuando envía el siguiente mensaje:

```
{"action": "sendmessage", "message": "hello, from Step Functions!"}
```

   La máquina de estados de Step Functions asociada a la ruta invoca una función de Lambda con el mensaje y la URL de devolución de llamada. La función de Lambda llama a la API de administración de API Gateway y envía el mensaje a todos los clientes conectados. Todos los clientes reciben el siguiente mensaje:

```
< hello, from Step Functions!
```

Ahora que ha probado la API de WebSocket, puede desconectarse de la API.

Para desconectarse de la API

- Pulse `CTRL+C` para desconectarse de la API.

Cuando un cliente se desconecta de la API, API Gateway invoca la ruta $disconnect de la API. La integración de Lambda para la ruta $disconnect de la API elimina el ID de conexión de DynamoDB.

Paso 7: limpiar

Para evitar costos innecesarios, elimine los recursos creados como parte de este tutorial Los siguientes pasos eliminan la pila y API de WebSocket de CloudFormation.

Para eliminar una API de WebSocket

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En la página de las API, seleccione la websocket-api.
3. Elija Action (Acciones), elija Delete (Eliminar)y, a continuación, confirme su elección.

Para eliminar una pila de CloudFormation

1. Abra la consola de CloudFormation en https://console.aws.amazon.com/cloudformation.
2. Seleccione su pila de CloudFormation.
3. Elija Delete (Eliminar) y, a continuación, confirme su elección.

Siguientes pasos

Puede automatizar la creación y la limpieza de todos los recursos de AWS involucrados en este tutorial. Para ver un ejemplo de una plantilla de CloudFormation que automatiza estas acciones para este tutorial, consulte ws-sfn.zip.

API de REST de API Gateway

Una API de REST en API Gateway es una colección de recursos y métodos que se integran con puntos de enlace de HTTP del backend, funciones de Lambda u otros servicios de AWS. Puede utilizar características de API Gateway para ayudarle con todos los aspectos del ciclo de vida de la API, desde la creación hasta el monitoreo de sus API de producción.

Las API de REST de API Gateway utilizan un modelo de solicitud/respuesta en el que un cliente envía una solicitud a un servicio y el servicio responde de forma sincrónica. Este tipo de modelo es adecuado para muchos tipos diferentes de aplicaciones que dependen de la comunicación sincrónica.

Temas

- Desarrollo de las API de REST en API Gateway
- Publicación de las API de REST para que los clientes las invoquen
- Optimización del rendimiento de las API de REST
- Distribución de sus API de REST a los clientes en API Gateway
- Protección de sus API de REST en API Gateway
- Supervisión de las API de REST en API Gateway

Desarrollo de las API de REST en API Gateway

En Amazon API Gateway, las API REST se crean como una colección de entidades programables conocidas como recursos de API Gateway. Por ejemplo, se utiliza un recurso RestApi para representar una API que puede contener una colección de entidades Resource.

Cada entidad `Resource` puede tener uno o más recursos Method. Un `Method` es una solicitud entrante enviada por el cliente y puede contener los siguientes parámetros de solicitud: un parámetro de ruta, un encabezado o un parámetro de cadena de consulta. Además, según el método HTTP, la solicitud puede contener un cuerpo. El método define cómo el cliente accede al `Resource` expuesto. Para integrar el `Method` con un punto de conexión de backend, también conocido como punto de conexión de integración, se crea un recurso de integración. Esto reenvía la solicitud entrante a un URI de punto de conexión de integración específico. Si es necesario, puede transformar los parámetros de la solicitud o el cuerpo de la solicitud para cumplir con los requisitos de backend o puede crear una integración de proxy, en la que API Gateway envía la solicitud completa en un

formato estandarizado al URI del punto de conexión de integración y luego envía directamente la respuesta al cliente.

Para las respuestas, puede crear un recurso MethodResponse para representar una respuesta recibida por el cliente y crear un recurso IntegrationResponse para representar la respuesta que devuelve el backend. Utilice una respuesta de integración para transformar los datos de respuesta del backend antes de devolver los datos al cliente o transferir la respuesta del backend tal y como está al cliente.

Recurso de ejemplo para una API de REST

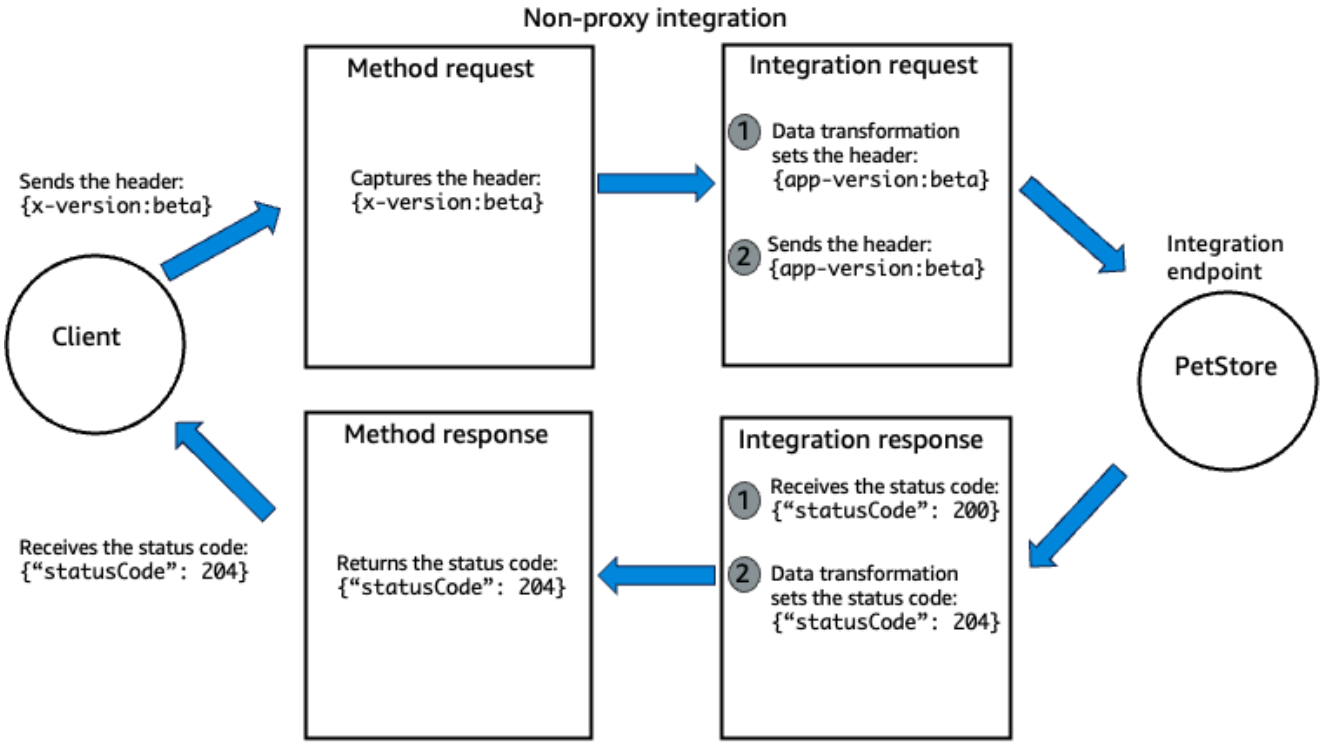
En el siguiente diagrama se muestra cómo API Gateway implementa este modelo de solicitud/respuesta para un proxy HTTP y una integración HTTP sin proxy para el recurso `GET /pets`. El cliente envía el encabezado `x-version:beta` a API Gateway y API Gateway envía el código de estado `204` al cliente.

En la integración sin proxy, API Gateway realiza transformaciones de datos para cumplir con los requisitos de backend, mediante la modificación de la solicitud de integración y la respuesta de integración. En una integración sin proxy, puede acceder al cuerpo en la solicitud del método, pero lo transforma en la solicitud de integración. Cuando el punto de conexión de integración devuelve una respuesta con un cuerpo, se accede a ella y se transforma en la respuesta de integración. No puede modificar el cuerpo en la respuesta del método.

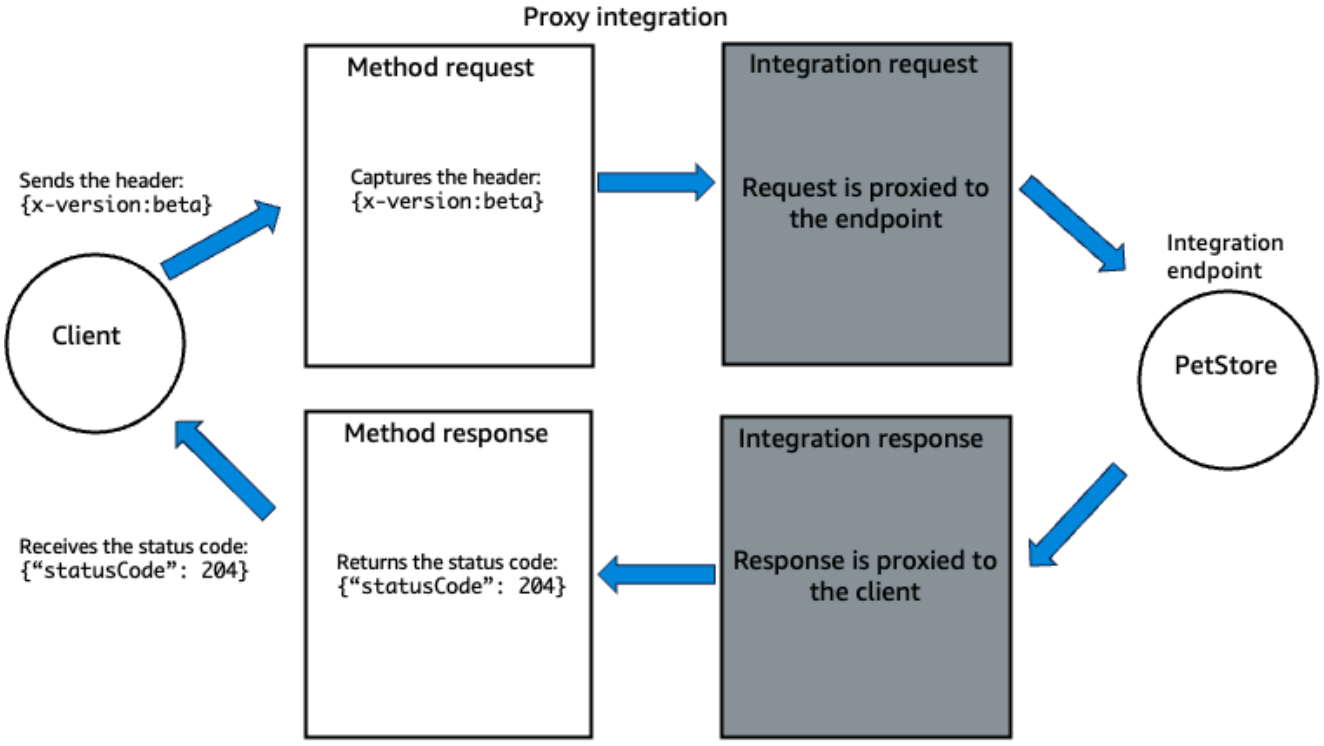
En la integración de proxy, el punto de conexión de integración modifica la solicitud y la respuesta. API Gateway no modifica la solicitud de integración ni la respuesta de integración y envía la solicitud entrante al backend tal cual.

Independientemente del tipo de integración, el cliente ha enviado una solicitud a API Gateway y API Gateway ha respondido de forma sincrónica.

Non-proxy integration



Proxy integration



Los siguientes registros de ejecución de ejemplo muestran lo que API Gateway registraría en el ejemplo anterior. Para mayor claridad, se han eliminado algunos valores y registros iniciales:

Non-proxy integration

```
Wed Feb 12 23:56:44 UTC 2025 : Starting execution for request: abcd-1234-5678
Wed Feb 12 23:56:44 UTC 2025 : HTTP Method: GET, Resource Path: /pets
Wed Feb 12 23:56:44 UTC 2025 : Method request path: {}
Wed Feb 12 23:56:44 UTC 2025 : Method request query string: {}
Wed Feb 12 23:56:44 UTC 2025 : Method request headers: {x-version=beta}
Wed Feb 12 23:56:44 UTC 2025 : Method request body before transformations:
Wed Feb 12 23:56:44 UTC 2025 : Endpoint request URI: http://petstore-demo-
endpoint.execute-api.com/petstore/pets
Wed Feb 12 23:56:44 UTC 2025 : Endpoint request headers: {app-version=beta}
Wed Feb 12 23:56:44 UTC 2025 : Endpoint request body after transformations:
Wed Feb 12 23:56:44 UTC 2025 : Sending request to http://petstore-demo-
endpoint.execute-api.com/petstore/pets
Wed Feb 12 23:56:45 UTC 2025 : Received response. Status: 200, Integration latency:
 123 ms
Wed Feb 12 23:56:45 UTC 2025 : Endpoint response headers: {Date=Wed, 12 Feb 2025
 23:56:45 GMT}
Wed Feb 12 23:56:45 UTC 2025 : Endpoint response body before transformations:
Wed Feb 12 23:56:45 UTC 2025 : Method response body after transformations: (null)
Wed Feb 12 23:56:45 UTC 2025 : Method response headers: {X-Amzn-Trace-Id=Root=1-
abcd-12345}
Wed Feb 12 23:56:45 UTC 2025 : Successfully completed execution
Wed Feb 12 23:56:45 UTC 2025 : Method completed with status: 204
```

Proxy integration

```
Wed Feb 12 23:59:42 UTC 2025 : Starting execution for request: abcd-1234-5678
Wed Feb 12 23:59:42 UTC 2025 : HTTP Method: GET, Resource Path: /pets
Wed Feb 12 23:59:42 UTC 2025 : Method request path: {}
Wed Feb 12 23:59:42 UTC 2025 : Method request query string: {}
Wed Feb 12 23:59:42 UTC 2025 : Method request headers: {x-version=beta}
Wed Feb 12 23:59:42 UTC 2025 : Method request body before transformations:
Wed Feb 12 23:59:42 UTC 2025 : Endpoint request URI: http://petstore-demo-
endpoint.execute-api.com/petstore/pets
Wed Feb 12 23:59:42 UTC 2025 : Endpoint request headers: { x-version=beta}
Wed Feb 12 23:59:42 UTC 2025 : Endpoint request body after transformations:
Wed Feb 12 23:59:42 UTC 2025 : Sending request to http://petstore-demo-
endpoint.execute-api.com/petstore/pets
```

```
Wed Feb 12 23:59:43 UTC 2025 : Received response. Status: 204, Integration latency:
 123 ms
Wed Feb 12 23:59:43 UTC 2025 : Endpoint response headers: {Date=Wed, 12 Feb 2025
 23:59:43 GMT}
Wed Feb 12 23:59:43 UTC 2025 : Endpoint response body before transformations:
Wed Feb 12 23:59:43 UTC 2025 : Method response body after transformations:
Wed Feb 12 23:59:43 UTC 2025 : Method response headers: {Date=Wed, 12 Feb 2025
 23:59:43 GMT}
Wed Feb 12 23:59:43 UTC 2025 : Successfully completed execution
Wed Feb 12 23:59:43 UTC 2025 : Method completed with status: 204
```

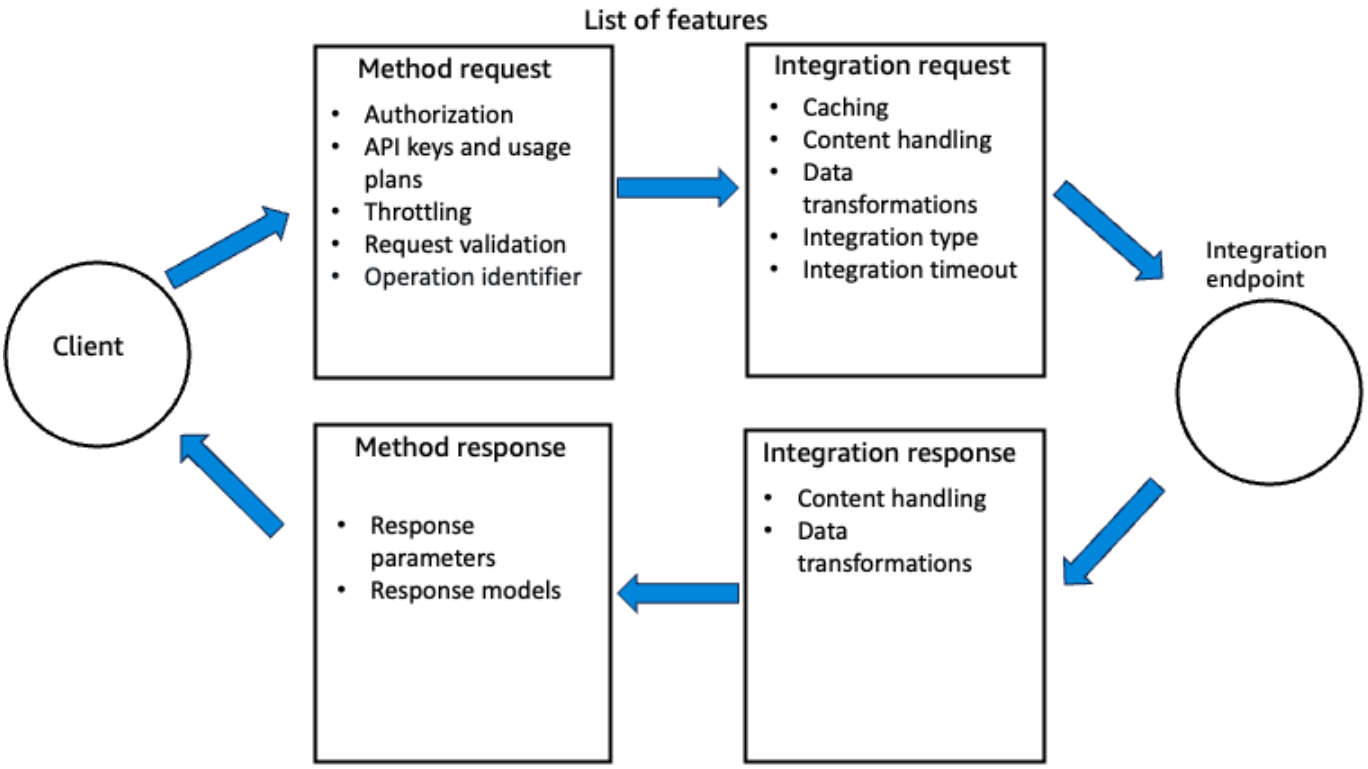
Para importar una API similar y probarla en la Consola de administración de AWS, consulte la API de ejemplo.

Características adicionales de la API de REST para el desarrollo

API Gateway admite características adicionales para el desarrollo de la API de REST. Por ejemplo, para ayudar a los clientes a comprender la API, puede proporcionar documentación para la API. Para ello, agregue un recurso DocumentationPart para una entidad de API compatible.

Para controlar el modo en que los clientes llaman a la API, utilice permisos de IAM, un autorizador de Lambda o un grupo de usuarios de Amazon Cognito. Para medir el uso de la API, configure planes de uso para limitar las solicitudes a la API. Puede habilitar estos elementos al crear o al actualizar la API.

En el siguiente diagrama se muestran las características disponibles para el desarrollo de la API de REST y dónde se configuran estas características en el modelo de solicitud/respuesta.



Para obtener una introducción sobre cómo crear una API, consulte the section called “Tutorial: Creación de una API de REST con una integración de proxy de Lambda”. Para obtener más información sobre las funciones de API Gateway que puede utilizar al desarrollar una API de REST, consulte los siguientes temas. Estos temas contienen información conceptual y procedimientos que puede realizar a través de la consola de API Gateway, la API de REST de API Gateway, la AWS CLI o uno de los AWS SDK.

Temas

- Tipos de punto de conexión para las API de REST en API Gateway
- Políticas de seguridad de las API de REST en API Gateway
- Tipos de direcciones IP para API de REST en API Gateway
- Métodos de API de REST en API Gateway
- Control y administración del acceso a las API de REST en API Gateway
- Integraciones para las API de REST en API Gateway
- Solicitud de la validación de las API de REST en API Gateway
- Transformaciones de datos para las API de REST en API Gateway
- Respuestas de Gateway para las API de REST en API Gateway

- CORS para las API de REST en API Gateway
- Tipos de medios binarios para las API de REST en API Gateway
- Invocación de las API de REST en API Gateway
- Desarrollo de API de REST mediante OpenAPI en API Gateway

Tipos de punto de conexión para las API de REST en API Gateway

Los tipos de punto de enlace de API hacen referencia al nombre de host de la API. El tipo de punto de conexión de la API puede ser optimizado para sistemas perimetrales, regional o privado, en función de dónde se origine la mayoría del tráfico de la API.

Punto de enlace de API optimizado para bordes

Un punto de conexión de API optimizada para límites normalmente dirige las solicitudes al punto de presencia (POP) de CloudFront más cercano, lo que podría ayudar en caso de que sus clientes estén en distintas ubicaciones geográficas. Este es el tipo de punto de enlace predeterminado para las API REST de API Gateway.

Las API optimizadas para sistemas perimetrales utilizan mayúsculas en la inicial de los nombres de los encabezados HTTP (por ejemplo, `Cookie`).

CloudFront ordena las cookies HTTP según su nombre antes de reenviar la solicitud al origen. Para obtener más información sobre el modo en que CloudFront procesa las cookies, consulte Almacenamiento en caché de contenido en función de cookies.

Cualquier nombre de dominio personalizado que se utilice con una API optimizada para sistemas perimetrales se aplicará a todas las regiones.

Puntos de enlace de API regionales

Un punto de conexión de la API regional está destinado a los clientes que se encuentran en la misma región. Cuando un cliente que se ejecuta en una instancia de EC2 llama a una API en la misma región o cuando una API tiene por objeto servir a una cantidad reducida de clientes con altas demandas, una API regional reduce la sobrecarga de conexión.

Para una API regional, el nombre de dominio personalizado es específico de la región en la que está implementada la API. Si implementa una API regional en varias regiones, esta puede tener el mismo nombre de dominio personalizado en todas las regiones. Puede utilizar dominios personalizados junto con Amazon Route 53 para realizar tareas como el direccionamiento basado en latencia.

Para obtener más información, consulte the section called “Configuración de un nombre de dominio personalizado regional” y the section called “Configuración de un nombre de dominio personalizado optimizado para la periferia para en API Gateway”.

Los puntos de enlace de API regionales transfieren todos los nombres de encabezado tal y como están.

> **Note**
>
> En los casos en que los clientes de la API están dispersos geográficamente, sigue teniendo sentido utilizar un punto de conexión de API regional, junto con su propia distribución de Amazon CloudFront para garantizar que API Gateway no asocia la API con distribuciones de CloudFront controladas por servicios. Para obtener más información acerca de este caso de uso, consulte la sección sobre cómo configurar API Gateway con mi propia distribución de CloudFront.

Puntos de enlace de API privados

Un punto de enlace de API privado es un punto de enlace de la API al que solo se pueda obtener acceso desde una Amazon Virtual Private Cloud (VPC) mediante un punto de enlace de la VPC de tipo interfaz, que es una interfaz de red de punto de enlace (ENI) que se crea en la VPC. Para obtener más información, consulte the section called “API de REST privadas”.

Los puntos de enlace de API privados transfieren todos los nombres de encabezado tal y como están.

Cambio del tipo de punto de conexión de una API pública o privada en API Gateway

Para cambiar el tipo de un punto de conexión de API, es preciso actualizar la configuración de la API. Puede cambiar un tipo de API existente a través de la consola de API Gateway, la AWS CLI o un AWS SDK para API Gateway. El tipo de punto de conexión no puede volver a cambiarse hasta que se complete el cambio actual, pero la API estará disponible.

Se admiten los siguientes cambios de tipo de puntos de conexión:

- De optimizado para bordes a regional o privado
- De regional a optimizado para bordes o privado
- De privado a regional

No puede transformar una API privada en una API optimizada para límites.

Si va a cambiar una API pública de optimizada para borde a regional o viceversa, tenga en cuenta que una API optimizada para borde puede comportarse de forma distinta a una API regional. Por ejemplo, una API optimizada para límites elimina el encabezado `Content-MD5`. Cualquier valor de hash MD5 transmitido al backend se puede expresar en un parámetro de cadena de solicitud o una propiedad del cuerpo. Sin embargo, la API regional transmite este encabezado, aunque puede reasignar el nombre de encabezado a otro nombre. Comprender estas diferencias le ayudará a decidir cómo actualizar una API optimizada para borde a una regional o cómo actualizar una API regional a una optimizada para borde.

Temas

- Uso de la consola de API Gateway para cambiar el tipo de punto de conexión de la API
- Uso de la AWS CLI para cambiar el tipo de punto de conexión de una API

Uso de la consola de API Gateway para cambiar el tipo de punto de conexión de la API

Para cambiar el tipo de punto de conexión de la API, siga uno de estos procedimientos:

Para convertir un punto de conexión público de regional u optimizado para límites y viceversa

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST.
3. Elija Configuración de la API.
4. En la sección Detalles de la API, elija Editar.
5. En Tipo de punto de conexión de la API, seleccione Optimizado para límites o Regional.
6. Seleccione Save changes (Guardar cambios).
7. Vuelva a implementar la API para que los cambios surtan efecto.

Conversión de un punto de conexión privado en un punto de conexión regional

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST.
3. Modifique la política de recursos de la API para eliminar cualquier mención a VPC o puntos de conexión de VPC, de modo que tanto las llamadas a la API que se realicen desde fuera de la VPC como las que se realicen desde dentro se ejecuten correctamente.

4. Elija Configuración de la API.
5. En la sección Detalles de la API, elija Editar.
6. En Tipo de punto de conexión de la API, seleccione Regional.
7. Seleccione Save changes (Guardar cambios).
8. Quite la política de recursos de la API.
9. Vuelva a implementar la API para que los cambios surtan efecto.

   Como está migrando el tipo de punto de conexión de privado a regional, API Gateway cambia el tipo de dirección IP a IPv4. Para obtener más información, consulte the section called “Tipos de direcciones IP para API de REST en API Gateway”.

Conversión de un punto de conexión regional en un punto de conexión privado

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST.
3. Cree una política de recursos que conceda acceso a la VPC o al punto de conexión de VPC. Para obtener más información, consulte ???.
4. Elija Configuración de la API.
5. En la sección Detalles de la API, elija Editar.
6. En Tipo de punto de conexión de la API, seleccione Privado.
7. (Opcional) Para ID del punto de conexión de VPC, seleccione los ID de punto de conexión de VPC que desea asociar a la API privada.
8. Seleccione Save changes (Guardar cambios).
9. Vuelva a implementar la API para que los cambios surtan efecto.

   Como está migrando el tipo de punto de conexión de regional a privado, API Gateway cambia el tipo de dirección IP a pila doble. Para obtener más información, consulte the section called “Tipos de direcciones IP para API de REST en API Gateway”.

Uso de la AWS CLI para cambiar el tipo de punto de conexión de una API

El siguiente comando update-rest-api permite actualizar una API optimizada para bordes a una API regional:

```
aws apigateway update-rest-api \
```

```
    --rest-api-id a1b2c3 \
    --patch-operations op=replace,path=/endpointConfiguration/types/EDGE,value=REGIONAL
```

La respuesta correcta tiene un código de estado `200 OK` y una carga similar a la siguiente:

```
{
    "createdDate": "2017-10-16T04:09:31Z",
    "description": "Your first API with Amazon API Gateway. This is a sample API that
 integrates via HTTP with our demo Pet Store endpoints",
    "endpointConfiguration": {
        "types": "REGIONAL"
    },
    "id": "a1b2c3",
    "name": "PetStore imported as edge-optimized"
}
```

El siguiente comando update-rest-api permite actualizar una API regional a una API optimizada para bordes:

```
aws apigateway update-rest-api \
    --rest-api-id a1b2c3 \
    --patch-operations op=replace,path=/endpointConfiguration/types/REGIONAL,value=EDGE
```

Dado que put-rest-api se utiliza para actualizar definiciones de API, no puede utilizarse para actualizar un tipo de punto de conexión de API.

Políticas de seguridad de las API de REST en API Gateway

Una política de seguridad es una combinación predefinida de versión mínima de TLS y conjuntos de cifrado que ofrece API Gateway. Cuando los clientes realizan un establecimiento de comunicación de TLS en la API o el nombre de dominio personalizado, la política de seguridad aplica la versión de TLS y el conjunto de cifrado aceptado por API Gateway. Las políticas de seguridad protegen las API y los nombres de dominio personalizados de los problemas de seguridad de red como, por ejemplo, la manipulación y el espionaje entre un cliente y el servidor.

API Gateway admite políticas de seguridad heredadas y políticas de seguridad mejoradas. `TLS_1_0` y `TLS_1_2` son políticas de seguridad heredadas. Utilice estas políticas de seguridad para compatibilidad con versiones anteriores. Cualquier política que comience con `SecurityPolicy_` es una política de seguridad mejorada. Utilice estas políticas para cargas de trabajo reguladas, una gobernanza avanzada o para utilizar la criptografía poscuántica. Al utilizar una política de seguridad

mejorada, también debe configurar el modo de acceso de punto de conexión para una mayor gobernanza. Para obtener más información, consulte the section called “Modo de acceso de punto de conexión”.

Cómo aplica API Gateway las políticas de seguridad

El siguiente ejemplo muestra cómo API Gateway aplica las políticas de seguridad mediante la política de seguridad `SecurityPolicy_TLS13_1_3_2025_09` como ejemplo.

La política de seguridad `SecurityPolicy_TLS13_1_3_2025_09` acepta el tráfico de TLS 1.3 y rechaza el tráfico de TLS 1.2 y TLS 1.0. Para el tráfico de TLS 1.3, la política de seguridad acepta los siguientes conjuntos de cifrado:

- `TLS_AES_128_GCM_SHA256`
- `TLS_AES_256_GCM_SHA384`
- `TLS_CHACHA20_POLY1305_SHA256`

API Gateway no acepta ningún otro conjunto de sistemas de cifrado. Por ejemplo, la política de seguridad rechazaría cualquier tráfico de TLS 1.3 que utilice el conjunto de cifrado de `AES128-SHA`. Para obtener más información acerca de los cifrados y versiones de TLS, consulte the section called “Políticas de seguridad admitidas”.

Para supervisar qué protocolo TLS y qué cifrado utilizan los clientes para acceder a la API Gateway, puede utilizar las variables de contexto `$context.tlsVersion` y `$context.cipherSuite` en los registros de acceso. Para obtener más información, consulte the section called “Supervisión”.

Modo de acceso de punto de conexión

El modo de acceso de punto de conexión es un parámetro adicional que debe especificar para cualquier API de REST o nombre de dominio personalizado que utilice una política de seguridad mejorada que comience por `SecurityPolicy_`. Esto se hace al crear el recurso o al cambiar la política de seguridad de una política heredada a una política mejorada.

Cuando el modo de acceso de punto de conexión está configurado como `STRICT`, cualquier solicitud a la API de REST o al nombre de dominio personalizado debe superar las siguientes comprobaciones:

- La solicitud debe provenir del mismo tipo de punto de conexión de API Gateway que el recurso. Puede provenir de un punto de conexión regional, optimizado para bordes o privado.

- Si utiliza un punto de conexión regional o privado, API Gateway usa la coincidencia de host de SNI. Si utiliza un punto de conexión optimizado para bordes, API Gateway cumple con la protección de domain fronting de CloudFront. Para obtener más información, consulte Domain fronting.

Si no se cumple alguna de estas condiciones, API Gateway rechaza la solicitud. Le recomendamos que utilice el modo de acceso de punto de conexión `STRICT` siempre que sea posible.

Para migrar una API o un nombre de dominio existentes para utilizar un modo de acceso estricto de punto de conexión, primero actualice la política de seguridad a una política de seguridad mejorada y mantenga el modo de acceso de punto de conexión configurado en `BASIC`. Tras validar los registros de tráfico y acceso, configure el modo de acceso de punto de conexión en `STRICT`. Al migrar el modo de acceso de punto de conexión de `STRICT` a `BASIC`, el punto de conexión no estará disponible durante unos 15 minutos a medida que se propaguen los cambios.

No debe configurar el modo de acceso de punto de conexión como `STRICT` para determinadas arquitecturas de aplicaciones, sino configurar el modo de acceso de punto de conexión como `BASIC`. En la siguiente tabla se muestran algunas arquitecturas de aplicaciones y una recomendación para que la API de REST o el nombre de dominio personalizado puedan utilizar el modo de acceso de punto de conexión `STRICT`.

Arquitectura	Migración sugerida
Uso de un punto de conexión de VPC para acceder a un nombre de dominio personalizado público.	Esta arquitectura utiliza tráfico de tipo cruzado entre puntos de conexión. Se recomienda que migre a the section called “Nombres de dominio personalizados para API privadas”.
Uso de cualquier método para invocar una API privada que no utilice un nombre de dominio personalizado o nombres de DNS privados.	Esta arquitectura crea una discordancia entre el encabezado del host y el SNI utilizado en el establecimiento de comunicación de TLS y no supera las restricciones de domain fronting de CloudFront. Le recomendamos que migre la VPC para usar un DNS privado.
Uso de la partición de dominios para distribuir contenido en varios dominios o subdominios.	Esta arquitectura crea una discordancia entre el encabezado del host y el SNI utilizado en el establecimiento de comunicación de TLS y no

Arquitectura	Migración sugerida
	supera las restricciones de domain fronting de CloudFront. Le recomendamos utilizar HTTP/2 y dejar de utilizar este antipatrón.

A continuación, se incluyen consideraciones que se deben tener a la hora de utilizar el modo de acceso de punto de conexión:

- Si el modo de acceso de punto de conexión de una API o un nombre de dominio es `STRICT`, no puede cambiar el tipo de punto de conexión. Para cambiar el tipo de punto de conexión, primero cambie el modo de acceso del punto de conexión a `BASIC`.
- Tras cambiar el modo de acceso del punto de conexión de `BASIC` a `STRICT`, API Gateway tarda 15 minutos en aplicar el modo de acceso estricto del punto de conexión.
- Al cambiar una política de seguridad de una política que comienza con `SecurityPolicy_` a una política heredada, debe desactivar el modo de acceso del punto de conexión a `""`.

Consideraciones

A continuación, se muestran consideraciones sobre las políticas de seguridad para las API de REST en API Gateway:

- Puede importar la política de seguridad en un archivo de definición de OpenAPI. Para obtener más información, consulte x-amazon-apigateway-security-policy.
- La API se puede asignar a un nombre de dominio personalizado con una política de seguridad diferente a la de la API. Al invocar ese nombre de dominio personalizado, API Gateway utiliza la política de seguridad del dominio personalizado para negociar el establecimiento de comunicación de TLS. Si desactiva el punto de conexión de la API predeterminado, esto podría afectar a la forma en que los intermediarios pueden invocar a la API.
- Si cambia la política de seguridad, esta tardará unos 15 minutos en completarse. Puede supervisar el `apiStatus` de la API. A medida que la API se actualice, el `apiStatus` será `UPDATING` y cuando se complete será `AVAILABLE`. Si el estado de la API es `UPDATING`, puede seguir invocándola.
- API Gateway admite políticas de seguridad en todas las API. Sin embargo, solo puede elegir una política de seguridad para las API de REST. API Gateway solo admite la política de seguridad `TLS_1_2` para las API HTTP o de WebSocket.

- No puede actualizar la política de seguridad de una API de `TLS_1_0` a `TLS_1_2`.
- Algunas políticas de seguridad admiten los conjuntos de cifrado ECDSA y RSA. Si utiliza este tipo de política con un nombre de dominio personalizado, los conjuntos de cifrado coinciden con el tipo de clave de certificado proporcionada por el cliente, ya sea RSA o ECDSA. Si utiliza este tipo de política con una API de REST, los conjuntos de cifrado coinciden con los conjuntos de cifrado compatibles con los tipos de certificados RSA.

Políticas de seguridad admitidas

En las siguientes tablas se describen las políticas de seguridad que se pueden especificar para cada tipo de punto de conexión de API de REST y tipo de nombre de dominio personalizado. Estas políticas le permiten controlar las conexiones entrantes. API Gateway solo admite TLS 1.2 en salida. Puede actualizar la política de seguridad de la API o nombre de dominio personalizado en cualquier momento.

Las políticas que contienen `FIPS` en el título son compatibles con el Estándar de procesamiento de la información federal (FIPS), que es un estándar de los gobiernos de EE. UU. y Canadá que especifica los requisitos de seguridad de los módulos criptográficos que protegen información confidencial. Para obtener más información, consulte Estándar de procesamiento de la información federal (FIPS) 140 en la página Conformidad de Seguridad en la nube de AWS.

Todas las políticas FIPS utilizan el módulo criptográfico AWS-LC validado para FIPS. Para obtener más información, consulte la página del módulo criptográfico AWS-LC en el sitio NIST Cryptographic Module Validation Program.

Las políticas que contienen `PQ` en el título utilizan la criptografía poscuántica (PQC) para implementar algoritmos híbridos de intercambio de claves para TLS a fin de garantizar la confidencialidad del tráfico contra las futuras amenazas de la computación cuántica.

Las políticas que contienen `PFS` en el título utilizan Perfect Forward Secrecy (PFS) para garantizar que las claves de sesión no se vean comprometidas.

Las políticas que contienen `FIPS` y `PQ` en su título admiten estas características.

Políticas de seguridad predeterminadas

Al crear una nueva API de REST o un dominio personalizado, se asigna al recurso una política de seguridad predeterminada. En la tabla siguiente se muestra la política de seguridad predeterminada para estos recursos.

Recurso	Nombre de la política de seguridad predeterminada
API regionales	TLS_1_0
API optimizadas para límites	TLS_1_0
API privadas	TLS_1_2
Dominio regional	TLS_1_2
Dominio optimizado para bordes	TLS_1_2
Dominio privado	TLS_1_2

Políticas de seguridad compatibles para las API regionales y privadas y los nombres de dominio personalizados

En la siguiente tabla se describen las políticas de seguridad que se pueden especificar para API regionales y privadas y nombres de dominio personalizados:

Política de seguridad	Versiones compatibles de TLS	Cifrados compatibles
SecurityPolicy_TLS13_1_3_2025_09	TLS1.3	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256

Política de seguridad	Versiones compatibles de TLS	Cifrados compatibles
SecurityPolicy_TLS13_1_3_FIPS_2025_09	TLS1.3	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384
SecurityPolicy_TLS13_1_2_FIPS_PFS_PQ_2025_09	TLS1.3 TLS1.2	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384

Política de seguridad	Versiones compatibles de TLS	Cifrados compatibles
SecurityPolicy_TLS13_1_2_PFS_PQ_2025_09	TLS1.3 TLS1.2	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384

Política de seguridad	Versiones compatibles de TLS	Cifrados compatibles
SecurityPolicy_TLS 13_1_2_PQ_2025_09	TLS1.3 TLS1.2	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-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

Política de seguridad	Versiones compatibles de TLS	Cifrados compatibles
SecurityPolicy_TLS13_1_2_2021_06	TLS1.3 TLS1.2	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384 • AES128-GCM-SHA256 • AES128-SHA256 • AES256-GCM-SHA384 • AES256-SHA256

Política de seguridad	Versiones compatibles de TLS	Cifrados compatibles
TLS_1_2	TLS1.3 TLS1.2	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384 • AES128-GCM-SHA256 • AES128-SHA256 • AES256-GCM-SHA384 • AES256-SHA256

Política de seguridad	Versiones compatibles de TLS	Cifrados compatibles
TLS_1_0	TLS1.3 TLS1.2 TLS1.1 TLS1.0	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 TLS1.0-TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES128-SHA • ECDHE-RSA-AES128-SHA • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384 • ECDHE-ECDSA-AES256-SHA

Política de seguridad	Versiones compatibles de TLS	Cifrados compatibles
		• ECDHE-RSA-AES256-SHA • AES128-GCM-SHA256 • AES128-SHA256 • AES128-SHA • AES256-GCM-SHA384 • AES256-SHA256 • AES256-SHA

Políticas de seguridad compatibles para las API optimizadas para bordes y los nombres de dominio personalizados

En la siguiente tabla se describen las políticas de seguridad que se pueden especificar para API optimizadas para bordes y nombres de dominio personalizados optimizados para bordes:

Nombre de política de seguridad	Versiones compatibles de TLS	Cifrados compatibles
SecurityPolicy_TLS13_2025_EDGE	TLS1.3	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256
SecurityPolicy_TLS12_PFS_2025_EDGE	TLS1.3 TLS1.2	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256

Nombre de política de seguridad	Versiones compatibles de TLS	Cifrados compatibles
		TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-ECDSA-CHACHA20-POLY1305 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-RSA-CHACHA20-POLY1305

Nombre de política de seguridad	Versiones compatibles de TLS	Cifrados compatibles
SecurityPolicy_TLS12_2018_EDGE	TLS1.3 TLS1.2	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-ECDSA-CHACHA20-POLY1305 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-RSA-CHACHA20-POLY1305 • ECDHE-RSA-AES256-SHA384

Nombre de política de seguridad	Versiones compatibles de TLS	Cifrados compatibles
TLS_1_2 (solo nombres de dominio personalizados optimizados para periferia)	TLS1.3 TLS1.2	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-ECDSA-CHACHA20-POLY1305 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-RSA-CHACHA20-POLY1305 • ECDHE-RSA-AES256-SHA384

Nombre de política de seguridad	Versiones compatibles de TLS	Cifrados compatibles
		• AES128-GCM-SHA256 • AES256-GCM-SHA384 • AES128-SHA256

Nombre de política de seguridad	Versiones compatibles de TLS	Cifrados compatibles
TLS_1_0	TLS1.3 TLS1.2 TLS1.1 TLS1.0	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 TLS1.0-TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES128-SHA • ECDHE-RSA-AES128-SHA • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384

Nombre de política de seguridad	Versiones compatibles de TLS	Cifrados compatibles
		• ECDHE-ECDSA-AES256-SHA • ECDHE-RSA-AES256-SHA • AES256-SHA256 • AES256-SHA

Nombre del cifrado OpenSSL y RFC

OpenSSL e IETF RFC 5246 utilizan nombres distintos para los mismos cifrados. En la siguiente tabla se asigna el nombre de OpenSSL al nombre de RFC para cada uno de los cifrados. Para obtener más información, consulte cifrados en la documentación de OpenSSL.

Nombre del cifrado de OpenSSL	Nombre del cifrado de RFC
TLS_AES_128_GCM_SHA256	TLS_AES_128_GCM_SHA256
TLS_AES_256_GCM_SHA384	TLS_AES_256_GCM_SHA384
TLS_CHACHA20_POLY1305_SHA256	TLS_CHACHA20_POLY1305_SHA256
ECDHE-RSA-AES128-GCM-SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
ECDHE-RSA-AES128-SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256

Nombre del cifrado de OpenSSL	Nombre del cifrado de RFC
ECDHE-RSA-AES128-SHA	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
ECDHE-RSA-AES256-GCM-SHA384	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
ECDHE-RSA-AES256-SHA384	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
ECDHE-RSA-AES256-SHA	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
AES128-GCM-SHA256	TLS_RSA_WITH_AES_128_GCM_SHA256
AES256-GCM-SHA384	TLS_RSA_WITH_AES_256_GCM_SHA384
AES128-SHA256	TLS_RSA_WITH_AES_128_CBC_SHA256
AES256-SHA	TLS_RSA_WITH_AES_256_CBC_SHA
AES128-SHA	TLS_RSA_WITH_AES_128_CBC_SHA
DES-CBC3-SHA	TLS_RSA_WITH_3DES_EDE_CBC_SHA

Cómo cambiar una política de seguridad

Puede cambiar la política de seguridad para la API. Si envía tráfico a las API a través del nombre de dominio personalizado, no es necesario que la API y el nombre de dominio personalizado tengan la misma política de seguridad. Al invocar ese nombre de dominio personalizado, API Gateway utiliza la política de seguridad del dominio personalizado para negociar el establecimiento de comunicación de TLS. Sin embargo, para mantener la coherencia, le recomendamos que utilice la misma política de seguridad para el nombre de dominio y la API personalizados.

Si cambia la política de seguridad, esta tardará unos 15 minutos en completarse. Puede supervisar el `apiStatus` de la API. A medida que la API se actualice, el `apiStatus` será `UPDATING` y cuando se complete será `AVAILABLE`. Cuando se actualice la API, puede seguir invocándola.

Consola de administración de AWS

Cambio de la política de seguridad de una API

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST.
3. Elija Configuración de la API y, a continuación, elija Editar.
4. Para Política de seguridad, seleccione una nueva política que comience con `SecurityPolicy_`.
5. Para Modo de acceso de punto de conexión, elija Estricto.
6. Seleccione Save changes (Guardar cambios).

   Vuelva a implementar la API para que los cambios se apliquen. Como ha cambiado el modo de acceso del punto de conexión a estricto, los cambios tardarán unos 15 minutos en propagarse por completo.

AWS CLI

El comando update-rest-api siguiente permite actualizar una API para utilizar la política de seguridad `SecurityPolicy_TLS13_1_3_2025_09`:

```
aws apigateway update-rest-api \
    --rest-api-id abcd1234 \
    --patch-operations '[
        {
            "op": "replace",
            "path": "/securityPolicy",
            "value": "SecurityPolicy_TLS13_1_3_2025_09"
        },
        {
            "op": "replace",
            "path": "/endpointAccessMode",
            "value": "STRICT"
        }
    ]'
```

El resultado será similar al siguiente:

```
{
    "id": "abcd1234",
    "name": "MyAPI",
    "description": "My API with a new security policy",
    "createdDate": "2025-02-04T11:47:06-08:00",
    "apiKeySource": "HEADER",
    "endpointConfiguration": {
        "types": [
            "REGIONAL"
        ],
        "ipAddressType": "dualstack"
    },
    "tags": {},
    "disableExecuteApiEndpoint": false,
    "securityPolicy": "SecurityPolicy_TLS13_1_3_2025_09",
    "endpointAccessMode": "STRICT"
    "rootResourceId": "efg456"
}
```

El comando update-rest-api siguiente actualiza una API que usaba una política de seguridad mejorada para utilizar la política de seguridad de `TLS_1_0`.

```
aws apigateway update-rest-api \
    --rest-api-id abcd1234 \
    --patch-operations '[
        {
            "op": "replace",
            "path": "/securityPolicy",
            "value": "TLS_1_0"
        },
        {
            "op": "replace",
            "path": "/endpointAccessMode",
            "value": ""
        }
    ]'
```

El resultado será similar al siguiente:

```
{
```

```
    "id": "abcd1234",
    "name": "MyAPI",
    "description": "My API with a new security policy",
    "createdDate": "2025-02-04T11:47:06-08:00",
    "apiKeySource": "HEADER",
    "endpointConfiguration": {
        "types": [
            "REGIONAL"
        ],
        "ipAddressType": "dualstack"
    },
    "tags": {},
    "disableExecuteApiEndpoint": false,
    "securityPolicy": "TLS_1_0",
    "rootResourceId": "efg456"
}
```


Tipos de direcciones IP para API de REST en API Gateway

Al crear una API, se especifica el tipo de direcciones IP que puede invocar la API. Puede elegir IPv4 para resolver las direcciones IPv4 para invocar la API o puede elegir pila doble para permitir que las direcciones IPv4 e IPv6 invoquen la API. Le recomendamos que establezca el tipo de dirección IP en pila doble para mitigar el agotamiento del espacio IP o para mejorar la postura de seguridad. Para obtener más información sobre los beneficios de un tipo de dirección IP de pila doble, consulte IPv6 en AWS.

Para restringir la API solo al tráfico IPv6, puede crear una política de recursos y restringir las direcciones IP de origen solo a los rangos de IPv6. Puede cambiar el tipo de dirección IP actualizando la configuración de la API. Este cambio se efectuará de forma inmediata y no es necesario que vuelva a implementar la API. Para obtener más información, consulte the section called “Ejemplo: Cómo denegar el tráfico a una API para una dirección o un rango de direcciones IP de origen”

Consideraciones para los tipos de direcciones IP

Es posible que las siguientes consideraciones afecten al uso de los tipos de direcciones IP:

- El tipo de dirección IP predeterminado para todas las API regionales y optimizadas para la periferia es IPv4.
- Las API privadas solo pueden tener un tipo de dirección IP de pila doble.

- Si cambia el tipo de dirección IP de una API existente de IPv4 a pila doble, confirme que todas las políticas que controlan el acceso a las API se hayan actualizado para tener en cuenta las llamadas de IPv6. Cuando cambia el tipo de dirección IP, el cambio surte efecto inmediatamente.
- Si migra el tipo de punto de conexión de una API de regional u optimizada para la periferia a privada, API Gateway cambia el tipo de dirección IP a pila doble. Para obtener más información, consulte the section called “Cambio del tipo de punto de conexión de una API pública o privada”.
- Si migra el tipo de punto de conexión de una API de privada a regional, debe configurar el tipo de dirección IP en pila doble. Una vez finalizada la migración del punto de conexión, puede cambiar el tipo de dirección IP a IPv4. Para obtener más información, consulte the section called “Cambio del tipo de punto de conexión de una API pública o privada”.
- La API se puede asignar a un nombre de dominio personalizado con un tipo de dirección IP diferente al de la API. Si desactiva el punto de conexión de la API predeterminado, esto podría afectar a la forma en que los intermediarios pueden invocar a la API.
- No puede usar un archivo de definición externo para configurar el tipo de dirección IP de la API.

Cambio del tipo de dirección IP de una API de REST

Puede cambiar el tipo de dirección IP actualizando la configuración de la API. Puede actualizar la configuración de la API mediante el uso de la Consola de administración de AWS, la AWS CLI, CloudFormation o un AWS SDK. Si cambia el tipo de dirección IP de la API, no es necesario volver a implementar la API para que los cambios surtan efecto. Antes de cambiar el tipo de dirección IP, confirme que las políticas que controlan el acceso a las API se hayan actualizado para tener en cuenta las llamadas de IPv6.

Consola de administración de AWS

Cambio del tipo de dirección IP de una API de REST

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST.
3. Elija Configuración de la API y, a continuación, elija Editar.
4. Para tipo de dirección IP, seleccione IPv4 o Pila doble.
5. Seleccione Save changes (Guardar cambios).

   El cambio en la configuración de la API se aplicará de forma inmediata.

AWS CLI

El siguiente comando update-rest-api actualiza una API para que tenga un tipo de dirección IP de pila doble:

```
aws apigateway update-rest-api \
    --rest-api-id abcd1234 \
    --patch-operations "op='replace',path='/endpointConfiguration/
ipAddressType',value='dualstack'"
```

El resultado será similar al siguiente:

```
{
    "id": "abcd1234",
    "name": "MyAPI",
    "description": "My API with a dualstack IP address type",
    "createdDate": "2025-02-04T11:47:06-08:00",
    "apiKeySource": "HEADER",
    "endpointConfiguration": {
        "types": [
            "REGIONAL"
        ],
        "ipAddressType": "dualstack"
    },
    "tags": {},
    "disableExecuteApiEndpoint": false,
    "rootResourceId": "efg456"
}
```

Métodos de API de REST en API Gateway

En API Gateway, un método de API incluye una solicitud de método y una respuesta de método. Se configura un método de API para definir lo que un cliente debería o debe hacer para enviar una solicitud de acceso al servicio en el backend y para definir las respuestas que recibe a cambio el cliente. Para la entrada, puede elegir los parámetros de solicitud de método, o bien una carga aplicable, para que el cliente proporcione los datos necesarios u opcionales en el tiempo de ejecución. Para la salida, se determina el código de estado de respuesta del método, los encabezados y el cuerpo aplicable como destinos a los que asignar los datos de respuesta del backend, antes de que se devuelvan al cliente. Para ayudar al desarrollador cliente a comprender los

comportamientos y los formatos de entrada y salida de su API, puede documentar la API y ofrecer mensajes de error adecuados para solicitudes no válidas.

Una solicitud de método de API es una solicitud HTTP. Para configurar la solicitud de método, debe configurar un método (o verbo) HTTP, la ruta a un recurso de API, los encabezados y los parámetros de cadenas de consulta aplicables. También configura una carga cuando el método HTTP es `POST`, `PUT` o `PATCH`. Por ejemplo, para recuperar una mascota con la API de muestra PetStore, define la solicitud de método de API de `GET /pets/{petId}`, donde `{petId}` es un parámetro de ruta que puede tomar un número en tiempo de ejecución.

```
GET /pets/1
Host: apigateway.us-east-1.amazonaws.com
...
```

Si el cliente especifica una ruta incorrecta, por ejemplo, `/pet/1` o `/pets/one` en lugar de `/pets/1`, se lanza una excepción.

Una respuesta de método de API es una respuesta HTTP con un código de estado determinado. Para una integración que no sea de proxy, debe configurar respuestas de método para especificar los destinos necesarios u opcionales de las asignaciones. Estos transforman los encabezados o el cuerpo de respuesta de la integración en encabezados o en cuerpo de respuesta de método. El mapeo puede ser tan sencillo como una transformación de identidad que pasa los encabezados o el cuerpo a través de la integración tal y como están. Por ejemplo, la siguiente respuesta de método `200` muestra un ejemplo de transferencia de una respuesta de integración correcta tal y como está.

```
200 OK
Content-Type: application/json
...

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

En principio, puede definir una respuesta de método correspondiente a una respuesta específica desde el backend. Normalmente, esto implica cualquier respuesta 2XX, 4XX y 5XX. Sin embargo, puede que no sea práctico, porque a menudo es posible que no sepa con antelación todas las respuestas que puede devolver un backend. En la práctica, puede designar una respuesta de

método como predeterminada para gestionar las respuestas desconocidas o no asignadas desde el backend. Es una buena práctica designar la respuesta 500 como predeterminada. En cualquier caso, debe configurar al menos una respuesta de método para integraciones que no sean de proxy. De lo contrario, API Gateway devuelve una respuesta de error 500 al cliente incluso cuando la solicitud se completa correctamente en el backend.

Para admitir un SDK con establecimiento inflexible de tipos para la API, como un SDK de Java, debe definir el modelo de datos de entrada para la solicitud de método y definir el modelo de datos de salida de la respuesta de método.

Requisitos previos

Antes de configurar un método de API, verifique lo siguiente:

- El método debe estar disponible en API Gateway. Siga las instrucciones en Tutorial: Creación de una API de REST con integración no de proxy HTTP.
- Si desea que el método se comunique con una función de Lambda, debe haber creado ya el rol de invocación de Lambda y el rol de ejecución de Lambda en IAM. También debe haber creado la función de Lambda con la que el método se comunicará en AWS Lambda. Para crear los roles y funciones, utilice las instrucciones de Creación de una función de Lambda para la integración de Lambda no de proxy de Elección de un tutorial de integración de AWS Lambda.
- Si desea que el método se comunique con una integración HTTP o de proxy HTTP, debe haber creado y tener acceso a la URL del punto de conexión HTTP con la que se comunicará el método.
- Compruebe que API Gateway admite los certificados de los puntos de conexión HTTP y proxy de HTTP. Para obtener más información, consulte Entidades de certificación compatibles con API Gateway para las integraciones HTTP y Proxy HTTP en API Gateway.

Temas

- Configurar una solicitud de método en API Gateway
- Configuración de una respuesta de método en API Gateway
- Configuración de un método con la consola de API Gateway

Configurar una solicitud de método en API Gateway

Configurar una solicitud de método implica realizar las siguientes tareas, después de la creación de un recurso RestApi:

1. Creación de una nueva API o elección de una entidad Resource de API existente.
2. Creación de un recurso Method de API que sea un verbo HTTP específico en el `Resource` nuevo o elegido de la API. Esta tarea se puede dividir aún más en las siguientes subtareas:
   - Añadido de un método HTTP a la solicitud de métodos
   - Configuración de parámetros de solicitudes
   - Definición de un modelo para el cuerpo de la solicitud
   - Aplicación de un esquema de autorización
   - Habilitación de la validación de la solicitud

Puede realizar estas tareas con los siguientes métodos:

- Consola de API Gateway
- Comandos de la AWS CLI (create-resource y put-method)
- Función del SDK de AWS (como, en Node.js, createResource y putMethod)
- API REST de API Gateway (resource:create y method:put).

Temas

- Configurar recursos de API
- Configurar un método HTTP
- Configurar parámetros de solicitud de método
- Configuración de un modelo de solicitud de método
- Configurar la autorización de solicitud de método
- Configurar la validación de solicitud de método

Configurar recursos de API

En una API de API Gateway, los recursos que se pueden dirigir se exponen como un árbol de entidades Resources de API, con el recurso de raíz (`/`) en la parte superior de la jerarquía. El recurso de raíz depende de la URL base de la API, que se compone del punto de conexión de la API y un nombre de etapa. En la consola de API Gateway, este URI base se denomina Invoke URI y se muestra en el editor de etapas de la API una vez que se implementa la API.

El punto de conexión de la API puede ser un nombre de host predeterminado o un nombre de dominio personalizado. El nombre de host predeterminado tiene el siguiente formato:

```
{api-id}.execute-api.{region}.amazonaws.com
```

En este formato, la *{api-id}* representa el identificador de la API que genera API Gateway. La variable *{region}* representa la región de AWS (por ejemplo, `us-east-1`) que eligió al crear la API. Un nombre de dominio personalizado es cualquier nombre fácil de recordar en un dominio de Internet válido. Por ejemplo, si ha registrado un dominio de Internet de `example.com`, `*.example.com` es un nombre de dominio personalizado válido. Para obtener más información, consulte configurar nombres de dominio personalizados.

Para la API de ejemplo de PetStore, el recurso de raíz (`/`) expone la tienda de mascotas. El recurso `/pets` representa la variedad de mascotas disponibles en la tienda. El `/pets/{petId}` expone una mascota individual de un identificador concreto (`petId`). El parámetro de la ruta de `{petId}` forma parte de los parámetros de solicitudes.

Para configurar un recurso de API, se elige un recurso existente como principal y, a continuación, se crea el recurso secundario bajo el recurso principal. Se empieza con el recurso raíz como principal, se añade un recurso a este principal, se añade otro recurso a este recurso secundario como nuevo principal, etc., a su identificador principal. A continuación, se añade el recurso designado al principal.

El siguiente comando get-resources permite recuperar todos los recursos de una API:

```
aws apigateway get-resources --rest-api-id apiId
```

En la API del ejemplo de PetStore, el resultado sería similar al siguiente:

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

Cada elemento enumera los identificadores del recurso (`id`) y, excepto el recurso raíz, su recurso principal inmediato (`parentId`), así como el nombre del recurso (`pathPart`). El recurso raíz es especial, ya que no tiene ninguno principal. Después de elegir un recurso como principal, use el siguiente comando para agregar un recurso secundario.

```
aws apigateway create-resource --rest-api-id apiId \
    --parent-id parentId \
    --path-part resourceName
```

Por ejemplo, para agregar comida para mascotas a la venta en el sitio web de PetStore, use el siguiente comando:

```
aws apigateway create-resource --rest-api-id a1b2c3 \
    --parent-id svzr2028x8 \
    --path-part food
```

El resultado será similar al siguiente:

```
{
    "path": "/food",
    "pathPart": "food",
    "id": "xdsvhp",
    "parentId": "svzr2028x8"
}
```

Utilizar un recurso proxy para simplificar la configuración de API

A medida que crece el negocio, el propietario de PetStore puede que decida añadir a la venta alimentos, juguetes y otros artículos relacionados con mascotas. Para sustentarlo, puede añadir

`/food`, `/toys` y otros recursos debajo del recurso raíz. Debajo de cada categoría de venta, es posible que también desee añadir más recursos, como por ejemplo `/food/{type}/{item}`, `/toys/{type}/{item}`, etc. Esto puede ser una tarea tediosa. Si decide añadir una capa intermedia `{subtype}` a las rutas de recursos para cambiar la jerarquía de rutas a `/food/{type}/{subtype}/{item}`, `/toys/{type}/{subtype}/{item}`, etc., los cambios afectarán a la configuración existente de la API. Para evitarlo, puede utilizar un recurso de proxy de API Gateway para exponer un conjunto de recursos de la API a la vez.

API Gateway define un recurso de proxy como un marcador de posición para que un recurso se especifique cuando se envíe la solicitud. Un recurso de proxy se expresa mediante un parámetro de ruta especial de `{proxy+}`, a menudo denominado parámetro de ruta expansiva. El signo + indica los recursos secundarios que lleva asociados. El marcador de posición `/parent/{proxy+}` equivale a cualquier recurso que coincida con el patrón de ruta de `/parent/*`. Puede usar cualquier cadena para el nombre del parámetro de ruta expansiva.

El siguiente comando create-resource permite crear un recurso de proxy bajo la raíz (`/{proxy+}`):

```
aws apigateway create-resource --rest-api-id apiId \
    --parent-id rootResourceId \
    --path-part {proxy+}
```

El resultado será similar al siguiente:

```
{
    "path": "/{proxy+}",
    "pathPart": "{proxy+}",
    "id": "234jdr",
    "parentId": "svzr2028x8"
}
```

Para el ejemplo de API de `PetStore`, puede utilizar `/{proxy+}` para representar tanto `/pets` como `/pets/{petId}`. Este recurso de proxy también puede hacer referencia a cualquier otro recurso (existente o que se vaya a añadir), como por ejemplo `/food/{type}/{item}`, `/toys/{type}/{item}`, etc., o `/food/{type}/{subtype}/{item}`, `/toys/{type}/{subtype}/{item}`, etc. El desarrollador del backend determina la jerarquía de recursos y el desarrollador cliente es responsable de comprenderlo. API Gateway simplemente transfiere lo que envíe el cliente al backend.

Una API puede tener más de un recurso de proxy. Por ejemplo, los siguientes recursos de proxy están permitidos dentro de una API, lo que supone que `/parent/{proxy+}` no está en el mismo segmento principal que `/parent/{child}/{proxy+}`.

```
/{proxy+}
/parent/{proxy+}
/parent/{child}/{proxy+}
```

Cuando un recurso de proxy tiene elementos secundarios que no son de proxy, los recursos secundarios se excluyen de la representación del recurso de proxy. En los ejemplos anteriores, `/{proxy+}` hace referencia a cualquier recurso bajo el recurso raíz, excepto los recursos `/parent[/*]`. En otras palabras, una solicitud de método realizada para un recurso específico prevalece sobre una solicitud de método realizada para un recurso genérico en el mismo nivel de la jerarquía de recursos.

En la siguiente tabla se muestra cómo API Gateway enruta las solicitudes a los siguientes recursos para la etapa `prod` de una API.

```
ANY /{proxy+}
GET /pets/{proxy+}
GET /pets/dog
```

Solicitud	Ruta seleccionada	Explicación
`GET https://api-id.execute-api. region.amazonaws.com/prod/pets/dog`	`GET /pets/dog`	La solicitud coincide completamente con este recurso.
`GET https://api-id.execute-api. region.amazonaws.com/prod/pets/cats`	`GET /pets/{proxy+}`	La variable de ruta expansiva `/pets/{proxy+}` captura esta solicitud.
`GET https://api-id.execute-`	`GET /{proxy+}`	La variable de ruta expansiva `/{proxy+}` captura esta solicitud.

Solicitud	Ruta seleccionada	Explicación
`api. region.amazonaws.com/prod/animals`		

Un recurso de proxy no puede tener cualquier recurso secundario. Un recurso de API después de `{proxy+}` es redundante y ambiguo. Los siguientes recursos de proxy no se permiten dentro de una API.

```
/{proxy+}/child
/parent/{proxy+}/{child}
/parent/{child}/{proxy+}/{grandchild+}
```

Configurar un método HTTP

Una solicitud de métodos de API se encapsula mediante el recurso Method de API Gateway. Para configurar la solicitud de método, primero debe iniciar el recurso `Method`, configurando al menos un método HTTP y un tipo de autorización en el método.

API Gateway, estrechamente asociado con el recurso de proxy, admite un método HTTP de `ANY`. Este método `ANY` representa cualquier método HTTP que se suministre en el tiempo de ejecución. Le permite utilizar una sola configuración de método de API para todos los métodos HTTP admitidos de `DELETE`, `GET`, `HEAD`, `OPTIONS`, `PATCH`, `POST` y `PUT`.

También puede configurar el método `ANY` en un recurso que no sea de proxy. Al combinar el método `ANY` con un recurso de proxy, obtiene una configuración de método de API único para todos los métodos compatibles con HTTP frente a cualquier recurso de una API. Además, el backend puede evolucionar sin que afecte a la configuración de API existente.

Antes de configurar un método de API, tenga en cuenta quién puede llamar al método. Establezca el tipo de autorización según su plan. Para un acceso abierto, establézcalo en `NONE`. Para utilizar permisos de IAM, establezca el tipo de autorización en `AWS_IAM`. Para utilizar una función de autorizador de Lambda, establezca esta propiedad en `CUSTOM`. Para utilizar un grupo de usuarios de Amazon Cognito, establezca el tipo de autorización en `COGNITO_USER_POOLS`.

El siguiente comando put-method permite crear una solicitud de método para el verbo `ANY` utilizando permisos de IAM para controlar el acceso.

```
aws apigateway put-method --rest-api-id vaz7da96z6 \
```

```
    --resource-id 6sxz2j \
    --http-method ANY \
    --authorization-type AWS_IAM
```

Para crear una solicitud de método de API con un tipo de autorización distinta, consulte the section called “Configurar la autorización de solicitud de método”.

Configurar parámetros de solicitud de método

Los parámetros de solicitud de método son una forma para que un cliente proporcione los datos de entrada o el contexto de ejecución necesarios para completar la solicitud de métodos. Un parámetro de método puede ser un método de ruta, un encabezado o un parámetro de cadena de consulta. Como parte de la configuración de solicitud de método, debe declarar los parámetros de solicitud necesarios para que estén disponibles para el cliente. Para la integración que no sea de proxy, puede traducir estos parámetros de solicitud en un formato que sea compatible con el requisito del backend.

Por ejemplo, para la solicitud de métodos `GET /pets/{petId}`, la variable de ruta `{petId}` es un parámetro de solicitud necesario. Puede declarar este parámetro de ruta cuando llame al comando `put-method` de la AWS CLI. El siguiente comando put-method permite crear un método con un parámetro de ruta obligatorio:

```
aws apigateway put-method --rest-api-id vaz7da96z6 \
    --resource-id rjkmth \
    --http-method GET \
    --authorization-type "NONE" \
    --request-parameters method.request.path.petId=true
```

Si un parámetro no es necesario, puede establecerlo en `false` en `request-parameters`. Por ejemplo, si el método `GET /pets` utiliza un parámetro de cadena de consulta opcional de `type` y un parámetro de encabezado opcional de `age`, puede declararlos usando el siguiente comando put-method:

```
aws apigateway put-method --rest-api-id vaz7da96z6 \
    --resource-id 6sxz2j \
    --http-method GET \
    --authorization-type "NONE" \
    --request-parameters
 method.request.querystring.type=false,method.request.header.age=false
```

En lugar de esta forma abreviada, puede utilizar una cadena de JSON para establecer el valor `request-parameters`:

```
'{"method.request.querystring.type":false,"method.request.header.age":false}'
```

Con esta configuración, el cliente puede consultar las mascotas por tipo:

```
GET /pets?type=dog
```

Y el cliente puede consultar qué perros son cachorros de la siguiente manera:

```
GET /pets?type=dog
age:puppy
```

Para obtener información sobre cómo asignar parámetros de solicitud de método a parámetros de solicitud de integración, consulte the section called “Integraciones”.

Configuración de un modelo de solicitud de método

Para un método de API que pueda tomar datos de entrada en una carga, puede utilizar un modelo. Un modelo se expresa en un esquema JSON, borrador 4 y describe la estructura de datos del cuerpo de la solicitud. Con un modelo, un cliente puede determinar cómo construir una carga de solicitud de métodos como entrada. Y lo que es más importante, API Gateway utiliza el modelo para validar una solicitud, generar un SDK e inicializar una plantilla de mapeo para configurar la integración en la consola de API Gateway. Para obtener información sobre cómo crear un modelo, consulte Comprensión de los modelos de datos.

En función de los tipos de contenido, una carga de método puede tener diferentes formatos. Un modelo se indexa frente al tipo de medio de la carga aplicada. API Gateway utiliza el encabezado de la solicitud `Content-Type` para determinar el tipo de contenido. Para configurar modelos de solicitud de método, añada pares de clave-valor con el formato `"`*`media-type`*`":"`*`model-name`*`"` al mapa `requestModels` cuando llame al comando AWS CLI de la `put-method`.

Para utilizar el mismo modelo independientemente del tipo de contenido, especifique `$default` como la clave.

Por ejemplo, para establecer un modelo en la carga útil de JSON de la solicitud del método `POST /pets` de la API de ejemplo de PetStore, puede usar el siguiente comando put-method:

```
aws apigateway put-method \
```

```
    --rest-api-id vaz7da96z6 \
    --resource-id 6sxz2j \
    --http-method POST \
    --authorization-type "NONE" \
    --request-models '{"application/json":"petModel"}'
```

Aquí, `petModel` es el valor de propiedad `name` de un recurso `Model` que describe una mascota. La definición del esquema real se expresa como un valor de cadena JSON de la propiedad `schema` del recurso `Model`.

En un SDK Java de la API, u otro tipo de SDK con establecimiento inflexible de tipos, los datos de entrada se forman como la clase `petModel` derivada de la definición del esquema. Con el modelo de solicitud, los datos de entrada en el SDK generado se forman en la clase `Empty`, que se deriva del modelo `Empty` predeterminado. En este caso, el cliente no puede crear una instancia de la clase de datos correcta para proporcionar la entrada necesaria.

Configurar la autorización de solicitud de método

Para controlar quién puede llamar al método de la API, puede configurar el tipo de autorización en el método. Puede utilizar este tipo para aplicar uno de los autorizadores admitidos, incluidos roles y políticas de IAM (`AWS_IAM`), un grupo de usuarios de Amazon Cognito (`COGNITO_USER_POOLS`), o un autorizador de Lambda basado en funciones de Lambda (`CUSTOM`).

Para utilizar permisos de IAM para autorizar el acceso al método de la API, establezca la propiedad de entrada `authorization-type` en **`AWS_IAM`**. Cuando esta opción está establecida, API Gateway verifica la firma del intermediario en la solicitud en función de las credenciales del intermediario. Si el usuario verificado tiene permiso para llamar al método, se acepta la solicitud. De lo contrario, rechaza la solicitud y el intermediario recibe una respuesta de error no autorizado. La llamada al método no se realiza correctamente a menos que el intermediario tenga permiso para invocar el método de la API. La siguiente política de IAM concede permiso al intermediario para llamar a cualquier método de API creado dentro de la misma Cuenta de AWS:

JSON

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

```
            "Action": [
                "execute-api:Invoke"
            ],
            "Resource": "arn:aws:execute-api:*:*:*"
        }
    ]
}
```

Para obtener más información, consulte the section called “Usar permisos de IAM”.

Actualmente, solo puedes conceder esta política a los usuarios, grupos y roles de la Cuenta de AWS del propietario de la API. Los usuarios de una Cuenta de AWS diferente pueden llamar a los métodos de la API solo si se les permite asumir un rol de la Cuenta de AWS del propietario de la API con los permisos necesarios para llamar a la acción `execute-api:Invoke`. Para obtener más información sobre los permisos entre cuentas, consulte Uso de roles de IAM.

Puede utilizar la AWS CLI, un AWS SDK o un cliente de API de REST, como Postman, que implementa la firma de Signature Version 4 (SigV4).

Para utilizar un autorizador de Lambda con el fin de autorizar el acceso al método de la API, establezca la propiedad de entrada `authorization-type` en `CUSTOM` y establezca la propiedad de entrada `authorizer-id` en el valor de propiedad `id` de un autorizador de Lambda que ya exista. El autorizador de Lambda al que se hace referencia puede ser del tipo `TOKEN` o `REQUEST`. Para obtener información acerca de cómo crear un autorizador de Lambda, consulte the section called “Uso de autorizadores Lambda”.

Para utilizar un grupo de usuarios de Amazon Cognito con el fin de autorizar el acceso al método de la API, establezca la propiedad de entrada `authorization-type` en `COGNITO_USER_POOLS` y establezca la propiedad de entrada `authorizer-id` en el valor de propiedad `id` del autorizador `COGNITO_USER_POOLS` que ya se creó. Para obtener información sobre la creación de un autorizador de grupo de usuarios de Amazon Cognito, consulte the section called “Uso de grupos de usuarios de Amazon Cognito como autorizador para una API REST”.

Configurar la validación de solicitud de método

Puede habilitar la validación de solicitud al configurar una solicitud de método de API. Primero debe crear un validador de solicitudes. El siguiente comando create-request-validator permite crear un validador de solicitud de solo cuerpo.

```
aws apigateway create-request-validator \
```

```
    --rest-api-id 7zw9uyk9kl \
    --name bodyOnlyValidator \
    --validate-request-body  \
    --no-validate-request-parameters
```

El resultado será similar al siguiente:

```
{
    "validateRequestParameters": false,
    "validateRequestBody": true,
    "id": "jgpyy6",
    "name": "bodyOnlyValidator"
}
```

Con este validador de solicitud, puede usar la validación de solicitudes como parte de la configuración de solicitudes del método. El siguiente comando put-method permite crear una solicitud de método que requiera que el cuerpo de la solicitud entrante coincida con el `PetModel` y tiene dos parámetros de solicitud que no son obligatorios:

```
aws apigateway put-method \
    --rest-api-id 7zw9uyk9kl \
    --resource-id xdsvhp \
    --http-method PUT \
    --authorization-type "NONE" \
    --request-parameters '{"method.request.querystring.type": false,
 "method.request.querystring.page":false}' \
    --request-models '{"application/json":"petModel"}' \
    --request-validator-id jgpyy6
```

Para incluir un parámetro de solicitud en la validación de la solicitud, debe establecer `validateRequestParameters` en `true` para el validador de solicitudes, y establecer el parámetro de solicitud específico en `true` en el comando `put-method`.

Configuración de una respuesta de método en API Gateway

Una respuesta de método de API encapsula el resultado de una solicitud de método de API que el cliente recibirá. Los datos de salida incluyen un código de estado HTTP, algunos encabezados y posiblemente un cuerpo.

Con las integraciones que no sean de proxy, los parámetros de respuesta especificados y el cuerpo se pueden asignar desde los datos de respuesta de integración asociados o se les pueden asignar

determinados valores estáticos según las asignaciones. Estas asignaciones se especifican en la respuesta de integración. La asignación puede ser una transformación idéntica que transfiere la respuesta de integración tal y como es.

Con una integración de proxy, API Gateway transfiere la respuesta del backend a la respuesta del método automáticamente. No es necesario configurar la respuesta del método de API. Sin embargo, con la integración proxy de Lambda, la función de Lambda debe devolver un resultado de este formato de salida para que API Gateway asigne correctamente la respuesta de integración a una respuesta de método.

Mediante programación, la configuración de la respuesta de método equivale a crear un recurso MethodResponse de API Gateway y a establecer las propiedades de statusCode, responseParameters y responseModels.

Al establecer códigos de estado para un método de API, debe elegir uno como predeterminado para gestionar cualquier respuesta de integración de un código de estado no previsto. Es razonable establecer `500` como predeterminado, ya que esto equivale a emitir respuestas que de otro modo no estarían asignadas como un error del servidor. Por motivos de instrucción, la consola de API Gateway establece la respuesta `200` como la opción predeterminada. Pero puede restablecerla en la respuesta `500`.

Para configurar una respuesta de método, debe haber creado la solicitud de método.

Configurar el código de estado de la respuesta de método

El código de estado de una respuesta de método define un tipo de respuesta. Por ejemplo, las respuestas de 200, 400 y 500 indican respuestas de éxito, de error del lado del cliente y de error del lado del servidor, respectivamente.

Para configurar un código de estado de respuesta de método, establezca la propiedad `statusCode` en un código de estado HTTP. El siguiente comando put-method-response permite crear la respuesta del método `200`.

```
aws apigateway put-method-response \
    --rest-api-id vaz7da96z6 \
    --resource-id 6sxz2j \
    --http-method GET \
    --status-code 200
```

Configurar parámetros de respuesta de métodos

Los parámetros de respuesta de método definen qué encabezados recibe el cliente en respuesta a la solicitud de método asociado. Los parámetros de respuesta también especifican un objetivo al que API Gateway asigna un parámetro de respuesta de integración, según los mapeos prescritos en la respuesta de integración del método de la API.

Para configurar los parámetros de respuesta de método, agregue al mapa `responseParameters` de `MethodResponse` pares de clave-valor con el formato `"{parameter-name}":"{boolean}"`. El siguiente comando put-method-response permite establecer el encabezado `my-header`.

```
aws apigateway put-method-response \
        --rest-api-id vaz7da96z6 \
        --resource-id 6sxz2j \
        --http-method GET \
        --status-code 200  \
        --response-parameters method.response.header.my-header=false
```

Configurar modelos de respuesta de método

Un modelo de respuesta de método define un formato del cuerpo de la respuesta del método. Es necesario configurar un modelo de respuesta de método al generar un SDK con establecimiento inflexible de tipos para la API. Garantiza que el resultado se emite en una clase apropiada en Java u Objective-C. En otros casos, la configuración de un modelo es opcional.

Antes de configurar el modelo de respuesta, primero debe crear el modelo en API Gateway. Para ello, puede llamar al comando `create-model`. El siguiente comando create-model permite crear un modelo `PetStorePet` para describir el cuerpo de la respuesta para la solicitud del método `GET /pets/{petId}`.

```
aws apigateway create-model \
    --rest-api-id vaz7da96z6 \
    --content-type application/json \
    --name PetStorePet \
    --schema '{ \
                "$schema": "http://json-schema.org/draft-04/schema#", \
                "title": "PetStorePet", \
                "type": "object", \
                "properties": { \
                  "id": { "type": "number" }, \
```

```
                    "type": { "type": "string" }, \
                    "price": { "type": "number" } \
                } \
            }'
```

El resultado se crea como un recurso `Model` de API Gateway.

Para configurar los modelos de respuesta de método para definir el formato de la carga, agregue el par de clave-valor "application/json":"PetStorePet" al mapa `requestModels` del recurso `MethodResponse`. El siguiente comando put-method-response permite crear un método de respuesta que utilice un modelo de respuesta para definir el formato de carga útil:

```
aws apigateway put-method-response \
    --rest-api-id vaz7da96z6 \
    --resource-id 6sxz2j \
    --http-method GET \
    --status-code 200  \
    --response-parameters method.response.header.my-header=false \
    --response-models '{"application/json":"PetStorePet"}'
```

Configuración de un método con la consola de API Gateway

Al crear un método mediante la consola de la API de REST, se configuran la solicitud de integración y la solicitud del método. De forma predeterminada, API Gateway crea la respuesta del método `200` para el método.

Las siguientes instrucciones muestran cómo editar la configuración de la solicitud de método y cómo crear respuestas de método adicionales para el método.

Temas

- Edición de una solicitud de método de API Gateway en la consola de API Gateway
- Configurar una respuesta de método de API Gateway en la consola de API Gateway

Edición de una solicitud de método de API Gateway en la consola de API Gateway

Estas instrucciones presuponen que ya ha creado la solicitud de método. Para obtener más información sobre cómo crear un método, consulte the section called “ Configuración de una solicitud de integración mediante la consola”.

1. En el panel Recursos, elija el método y, a continuación, elija la pestaña Solicitud de método.

2. En la sección Configuración de solicitud de método, elija Editar.
3. En Autorización, seleccione un autorizador disponible.

   a. Para habilitar el acceso abierto al método a cualquier usuario, elija Ninguno. Este paso se puede omitir si el ajuste predeterminado no se ha cambiado.

   b. Para utilizar permisos de IAM para controlar el acceso de clientes al método, seleccione `AWS_IAM`. Con esta opción, solo los usuarios de los roles de IAM con la política correcta de IAM asociada pueden llamar a este método.

      Para crear el rol de IAM, especifique una política de acceso con un formato como el siguiente:

      JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:us-east-1:111111111111:aaabbb/*/GET/"
      ]
    }
  ]
}
```

      En esta política de acceso, *arn:aws:execute-api:us-east-1:111111111111:aaabbb/*/GET/* es el ARN del método. Puede encontrar el ARN del método mediante la selección del método en la página de Recursos. Para obtener más información acerca de cómo establecer los permisos de IAM, consulte Control del acceso a una API de REST con permisos de IAM.

      Para crear un rol de IAM, puede adaptar las instrucciones del siguiente tutorial, ???.

   c. Para usar un autorizador de Lambda, seleccione un token o un autorizador de solicitudes. Cree un autorizador de Lambda para que esta opción se muestre en el menú desplegable.

Para obtener más información sobre cómo crear un autorizador de Lambda, consulte Uso de autorizadores Lambda de API Gateway.

d. Para utilizar un grupo de usuarios de Amazon Cognito, elija un grupo de usuarios disponible en Cognito user pool authorizers (Autorizadores de grupos de usuarios de Cognito). Cree un grupo de usuarios en Amazon Cognito y un autorizador de grupo de usuarios de Amazon Cognito en API Gateway para que esta opción se muestre en el menú desplegable. Para obtener información sobre cómo crear un autorizador de grupo de usuarios de Amazon Cognito, consulte Control del acceso a las API de REST con grupos de usuarios de Amazon Cognito como autorizador.

4. Para especificar la validación de la solicitud, seleccione un valor en el menú desplegable Validador de solicitudes. Para desactivar la validación de solicitudes, seleccione Ninguna. Para obtener más información acerca de cada opción, consulte Solicitud de la validación de las API de REST en API Gateway.

5. Seleccione Clave de API obligatoria para que se requiera una clave de API. Cuando se habilita, las claves de API se utilizan en planes de uso para limitar el tráfico del cliente.

6. De forma opcional, para asignar un nombre de operación en un SDK de Java de esta API, generado por API Gateway, ingrese un nombre en Nombre de operación. Por ejemplo, para la solicitud de método de `GET /pets/{petId}`, el nombre de operación de SDK de Java correspondiente es por defecto `GetPetsPetId`. Este nombre se crea a partir del verbo HTTP del método (`GET`) y los nombres de variables de la ruta de recursos (`Pets` y `PetId`). Si establece el nombre de operación como `getPetById`, el nombre de operación de SDK pasa a ser `GetPetById`.

7. Para añadir un parámetro de cadena de consulta al método, haga lo siguiente:

   a. Elija Parámetros de cadenas de consulta de URL y, a continuación, elija Agregar cadena de consulta.

   b. En Nombre, escriba el nombre del parámetro de cadena de consulta.

   c. Seleccione Obligatorio si el parámetro de cadena de consulta recién creado debe utilizarse para la validación de solicitudes. Para obtener más información sobre la validación de solicitud, consulte Solicitud de la validación de las API de REST en API Gateway.

   d. Seleccione Almacenamiento en caché si el parámetro de cadena de consulta recién creado va a utilizarse como parte de una clave de almacenamiento en caché. Para obtener más información acerca del almacenamiento en caché, consulte Usar parámetros de método o integración como claves de caché para indexar las respuestas almacenadas en caché.

Para eliminar el parámetro de cadena de consulta, seleccione Eliminar.

8. Para añadir un parámetro de encabezado al método, haga lo siguiente:

   a. Elija Encabezados de solicitud HTTP y, a continuación, elija Agregar encabezado.

   b. En Nombre, ingrese el nombre del encabezado.

   c. Seleccione Obligatorio si el encabezado recién creado debe utilizarse para la validación de solicitudes. Para obtener más información sobre la validación de solicitud, consulte Solicitud de la validación de las API de REST en API Gateway.

   d. Seleccione Almacenamiento en caché si el encabezado recién creado va a utilizarse como parte de una clave de almacenamiento en caché. Para obtener más información acerca del almacenamiento en caché, consulte Usar parámetros de método o integración como claves de caché para indexar las respuestas almacenadas en caché.

   Para eliminar el encabezado, elija Eliminar.

9. Para declarar el formato de carga de una solicitud de método con el verbo HTTP `POST`, `PUT` o `PATCH`, elija Cuerpo de la solicitud y realice lo siguiente:

   a. Elija Add model (Añadir modelo).

   b. En Content-type, escriba un tipo de MIME (por ejemplo, `application/json`).

   c. En Modelo, seleccione un modelo en el menú desplegable. Los modelos disponibles actualmente para la API incluyen los modelos predeterminados `Empty` y `Error`, así como cualquier modelo que haya creado y agregado a la colección Models de la API. Para obtener más información acerca de la creación de un modelo, consulte Modelos de datos para las API de REST.

   > **Note**
   >
   > El modelo es útil para informar al cliente del formato de datos esperado de una carga. Resulta útil para generar una plantilla de asignación de esqueleto. Es importante para generar un SDK con establecimiento inflexible de tipos de la API en idiomas como Java, C #, Objective-C y Swift. Solo es necesario si se habilita la validación de solicitudes frente a la carga.

10. Seleccione Save.

Configurar una respuesta de método de API Gateway en la consola de API Gateway

Un método de API puede tener una o más respuestas. Cada respuesta se indexa mediante su código de estado HTTP. De forma predeterminada, la consola de API Gateway agrega la respuesta `200` a las respuestas de método. Puede modificarla, por ejemplo, para que el método devuelva `201` en su lugar. Puede añadir otras respuestas, por ejemplo, `409` para la denegación de acceso y `500` para las variables de etapa sin inicializar utilizadas.

Para utilizar la consola de API Gateway para modificar, eliminar o agregar una respuesta a un método de API, siga estas instrucciones.

1. En el panel de Recursos, elija el método y, a continuación, elija la pestaña Respuesta de método. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña.
2. En la sección Configuración de respuesta del método, elija Crear respuesta.
3. Para Código de estado HTTP, ingrese un código de estado HTTP como `200`, `400` o `500`.

   Si una respuesta devuelta del backend no tiene una respuesta de método correspondiente definida, API Gateway no puede devolver la respuesta al cliente. En su lugar, devuelve una respuesta de error `500 Internal server error`.
4. Elija Agregar encabezado.
5. En Nombre del encabezado, escriba un nombre.

   Para devolver un encabezado del backend al cliente, agregue el encabezado en la respuesta del método.
6. Elija Agregar modelo para definir un formato del cuerpo de la respuesta del método.

   Ingrese el tipo de medio de la carga de respuesta de Tipo de contenido y elija un modelo en el menú desplegable Modelos.
7. Seleccione Save.

Para modificar una respuesta existente, navegue hasta la respuesta de su método y, a continuación, elija Editar. Para cambiar el Código de estado HTTP, elija Eliminar y cree una nueva respuesta de método.

Para cada respuesta que devuelve el backend, debe tener una respuesta compatible configurada como respuesta de método. Sin embargo, los encabezados de respuesta de método de configuración y el modelo de carga son opcionales, a menos que asigne el resultado del backend

a la respuesta de método antes de volver al cliente. Además, un modelo de carga de respuesta de método es importante si está generando un SDK con establecimiento inflexible de tipos para su API.

Control y administración del acceso a las API de REST en API Gateway

API Gateway admite varios mecanismos para controlar y administrar el acceso a la API.

Puede utilizar os siguientes mecanismos para realizar la autenticación y autorización:

- Gracias a las políticas de recursos, puede crear políticas basadas en recursos para permitir o denegar el acceso a sus API y métodos desde determinadas direcciones IP de origen o determinados puntos de enlace de la VPC. Para obtener más información, consulte the section called “Uso de políticas de recursos de API Gateway”.
- Los roles y las políticas estándar de AWS IAM ofrecen controles de acceso flexibles y robustos que se pueden aplicar a toda una API o a métodos individuales. Puede usar roles y políticas de IAM para controlar quién puede crear y administrar sus API, así como quién puede invocarlas. Para obtener más información, consulte the section called “Usar permisos de IAM”.
- Las etiquetas de IAM se pueden utilizar con políticas de IAM para controlar el acceso. Para obtener más información, consulte the section called “Control de acceso basado en atributos”.
- Las políticas de punto de enlace para los puntos de enlace de la VPC de interfaz le permiten asociar las políticas de recursos de IAM a puntos de enlace de interfaz de la VPC para mejorar la seguridad de sus API privadas. Para obtener más información, consulte the section called “Uso de políticas de punto de conexión de VPC para API privadas”.
- Los autorizadores Lambda son funciones de Lambda que controlan el acceso a los métodos de la API REST utilizando la autenticación de token al portador, así como la información descrita por los encabezados, rutas, cadenas de consulta, variables de etapa o parámetros de solicitud de variables contextuales. Los autorizadores Lambda se utilizan para controlar quién puede invocar los métodos REST API. Para obtener más información, consulte the section called “Uso de autorizadores Lambda”.
- Los grupos de usuarios de Amazon Cognito permiten crear soluciones personalizables de autenticación y autorización para las API REST. Los grupos de usuarios de Amazon Cognito se utilizan para controlar quién puede invocar los métodos de la API REST. Para obtener más información, consulte the section called “Uso de grupos de usuarios de Amazon Cognito como autorizador para una API REST”.

Puede utilizar los siguientes mecanismos para realizar otras tareas relacionadas con el control de acceso:

- El uso compartido de recursos entre orígenes (CORS) permite controlar la forma en que la API REST responde a las solicitudes de recursos entre dominios. Para obtener más información, consulte the section called “CORS”.
- Los certificados SSL del lado del cliente pueden utilizarse para verificar que las solicitudes HTTP dirigidas a su sistema backend proceden de API Gateway. Para obtener más información, consulte the section called “Certificados generados por API Gateway”.
- AWS WAF se puede utilizar para proteger la API de API Gateway frente a ataques web comunes. Para obtener más información, consulte the section called “AWS WAF”.

Puede utilizar los siguientes mecanismos para rastrear y limitar el acceso concedido a los clientes autorizados:

- Los planes de uso permiten proporcionar claves de API a sus clientes y, después, realizar un seguimiento y limitar el uso de etapas y métodos de API para cada clave de API. Para obtener más información, consulte the section called “Planes de uso”.

Control del acceso a una API de REST con políticas de recursos de API Gateway

Las políticas de recursos de Amazon API Gateway son documentos de política JSON que se asocian a una API para controlar si una entidad principal especificada (por lo general, un rol o un grupo de IAM) puede invocar la API. Puede utilizar las políticas de recursos de API Gateway para permitir la invocación segura de la API por parte de:

- Los usuarios de una cuenta de AWS especificada.
- Los intervalos de direcciones IP o bloques de CIDR especificados.
- Las nubes privadas virtuales (VPC) o los puntos de conexión de VPC (de cualquier cuenta) especificados.

Puede asociar una política de recursos para cualquier tipo de punto de conexión de la API en API Gateway mediante la Consola de administración de AWS, la CLI de AWS o los AWS SDK. Para API privadas, puede utilizar las políticas de recursos junto con las políticas de punto de conexión de VPC para controlar qué entidades principales tienen acceso a qué recursos y acciones. Para obtener más información, consulte the section called “Uso de políticas de punto de conexión de VPC para API privadas”.

Las políticas de recursos de API Gateway son diferentes de las políticas basadas en entidades de IAM. Las políticas basadas en identidades de IAM se asocian a usuarios, grupos o roles de IAM y definen qué acciones pueden realizar esas identidades y en qué recursos. Las políticas de recursos de API Gateway están asociadas a recursos. Puede utilizar las políticas de recursos de API Gateway junto con las políticas de IAM. Para obtener más información, consulte Políticas basadas en identidad y políticas basadas en recursos.

Temas

- Información general del lenguaje de políticas de acceso para Amazon API Gateway
- Cómo afectan las políticas de recursos de API Gateway al flujo de trabajo de autorización
- Ejemplos de políticas de recursos de API Gateway
- Creación y asociación de una política de recursos de API Gateway a una API
- AWSClaves de condición de que se pueden utilizar en las políticas de recursos de API Gateway

Información general del lenguaje de políticas de acceso para Amazon API Gateway

Esta página describe los elementos básicos utilizados en las políticas de recursos de Amazon API Gateway.

Las políticas de recursos se especifican utilizando la misma sintaxis que para las políticas de IAM. Para obtener información completa acerca del lenguaje de políticas, consulte Información general de políticas de IAM y Referencia de políticas de AWS Identity and Access Management en la Guía del usuario de IAM.

Para obtener información sobre cómo decide un servicio de AWS si se permite o se deniega una solicitud determinada, consulte Determinar si se permite o deniega una solicitud.

Elementos comunes en una política de acceso

Una política de recursos contiene los siguientes elementos básicos:

- Recursos: las API son los recursos de Amazon API Gateway para los que puede conceder o denegar permisos. En una política, se usa el nombre de recurso de Amazon (ARN) para identificar el recurso. También puede utilizar sintaxis abreviada, que API Gateway expande automáticamente al ARN completo al guardar una política de recursos. Para obtener más información, consulte Ejemplos de políticas de recursos de API Gateway.

Para conocer el formato del elemento `Resource` completo, consulte Formato de Resource de permisos para ejecutar la API en API Gateway.

- Acciones: para cada recurso, Amazon API Gateway admite un conjunto de operaciones. Con las palabras clave de acción puede identificar las operaciones del recurso que desea permitir o denegar.

  Por ejemplo, el permiso `execute-api:Invoke` autorizará al usuario a invocar una API previa solicitud del cliente.

  Para conocer el formato del elemento `Action`, consulte Formato de Action de permisos para ejecutar la API en API Gateway.
- Efecto: el efecto que se obtendrá cuando el usuario solicite la acción específica. Puede ser `Allow` o `Deny`. También puede denegar de forma explícita el acceso a un recurso para asegurarse de que un usuario no obtenga acceso a él, aunque otra política se lo conceda.

  > **Note**
  >
  > "Denegar de forma implícita" es lo mismo que "denegar de forma predeterminada". Una "negación implícita" es diferente de una "negación explícita". Para obtener más información, consulte Diferencia entre denegar de forma predeterminada y la denegación explícita.
- Principal (Entidad principal): la cuenta o el usuario con permiso de acceso a las acciones y los recursos en la instrucción. En una política de recursos, la entidad principal es el usuario o la cuenta que recibe este permiso.

La política de recursos del ejemplo siguiente muestra los elementos comunes de política anteriores. La política concede acceso a la API bajo el *`account-id`* especificado en la *`región`* especificada a cualquier usuario cuya dirección IP de origen está en el bloque de direcciones *`123.4.5.6/24`*. La política deniega todo el acceso a la API si la IP de origen del usuario no está dentro del rango.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "arn:aws:execute-api:us-east-1:111111111111:*"
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "arn:aws:execute-api:us-east-1:111111111111:*",
            "Condition": {
                "NotIpAddress": {
                    "aws:SourceIp": "123.4.5.6/24"
                }
            }
        }
    ]
}
```

Cómo afectan las políticas de recursos de API Gateway al flujo de trabajo de autorización

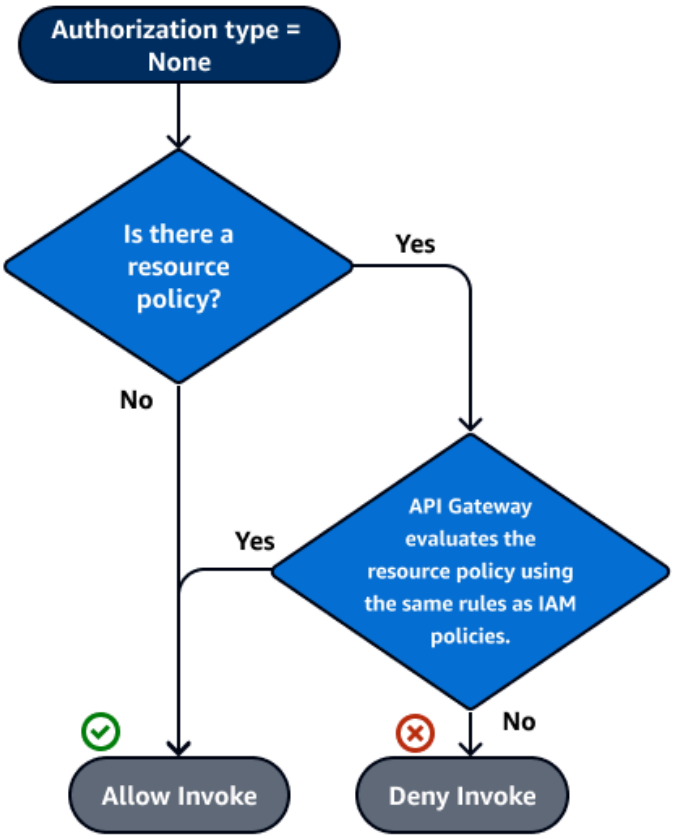
Cuando API Gateway evalúa la política de recursos asociada a la API, el resultado depende del tipo de autenticación que se haya definido para la API, tal y como se muestra en los diagramas de flujo de las siguientes secciones.

Temas

- Solo política de recursos de API Gateway
- Autorizador de Lambda y política de recursos
- Autenticación de IAM y política de recursos
- Política de recursos y autenticación de Amazon Cognito
- Tablas de resultados de evaluación de políticas

Solo política de recursos de API Gateway

En este flujo de trabajo, una política de recursos de API Gateway está asociada a la API, pero no se ha definido ningún tipo de autenticación para la API. Para evaluar la política ay que buscar un permiso explícito en función de los criterios de entrada del intermediario. Una denegación implícita o cualquier denegación explícita provoca la denegación del intermediario.



El siguiente ejemplo muestra dicha política.

JSON

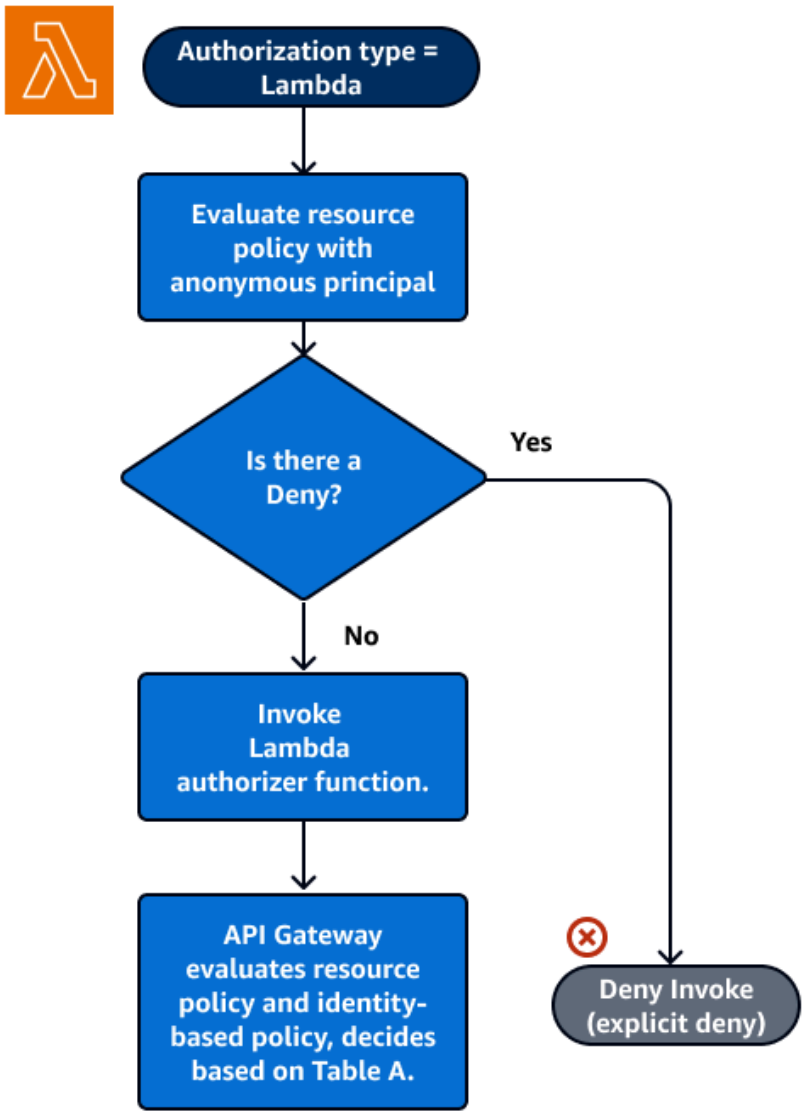
```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "arn:aws:execute-api:us-east-1:111111111111:api-id/",
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": ["192.0.2.0/24", "198.51.100.0/24" ]
                }
            }
        }
    ]
```

```
}
```

Autorizador de Lambda y política de recursos

En este flujo de trabajo, un autorizador de Lambda está configurado para la API, además de una política de recursos. La política de recursos se evalúan en dos fases. Antes de llamar al autorizador de Lambda, API Gateway primero evalúa la política y comprueba si hay denegaciones explícitas. Si se encuentran, se deniega inmediatamente el acceso al intermediario. De lo contrario, se llamada al autorizador de Lambda, que devuelve un documento de política, que se evalúa junto con la política de recursos. Si el autorizador utiliza el almacenamiento en caché, es posible que API Gateway devuelva el documento de política almacenado en caché. El resultado se determina en función de la Tabla A.

La siguiente política de recursos de ejemplo solo permite llamadas desde el punto de enlace de la VPC cuyo ID de punto de enlace de la VPC sea *vpce-1a2b3c4d*. Durante la evaluación "previa a la autenticación"·, solo se permiten las llamadas procedentes del punto de enlace de la VPC que se indican l el ejemplo para seguir adelante y evaluar el autorizador de Lambda. Todas las llamadas restantes se bloquean. Este flujo de trabajo de autorización es el mismo si utiliza un nombre de dominio personalizado para una API privada.



JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "arn:aws:execute-api:us-east-1:111111111111:api-id/"
            ],
            "Condition" : {
```

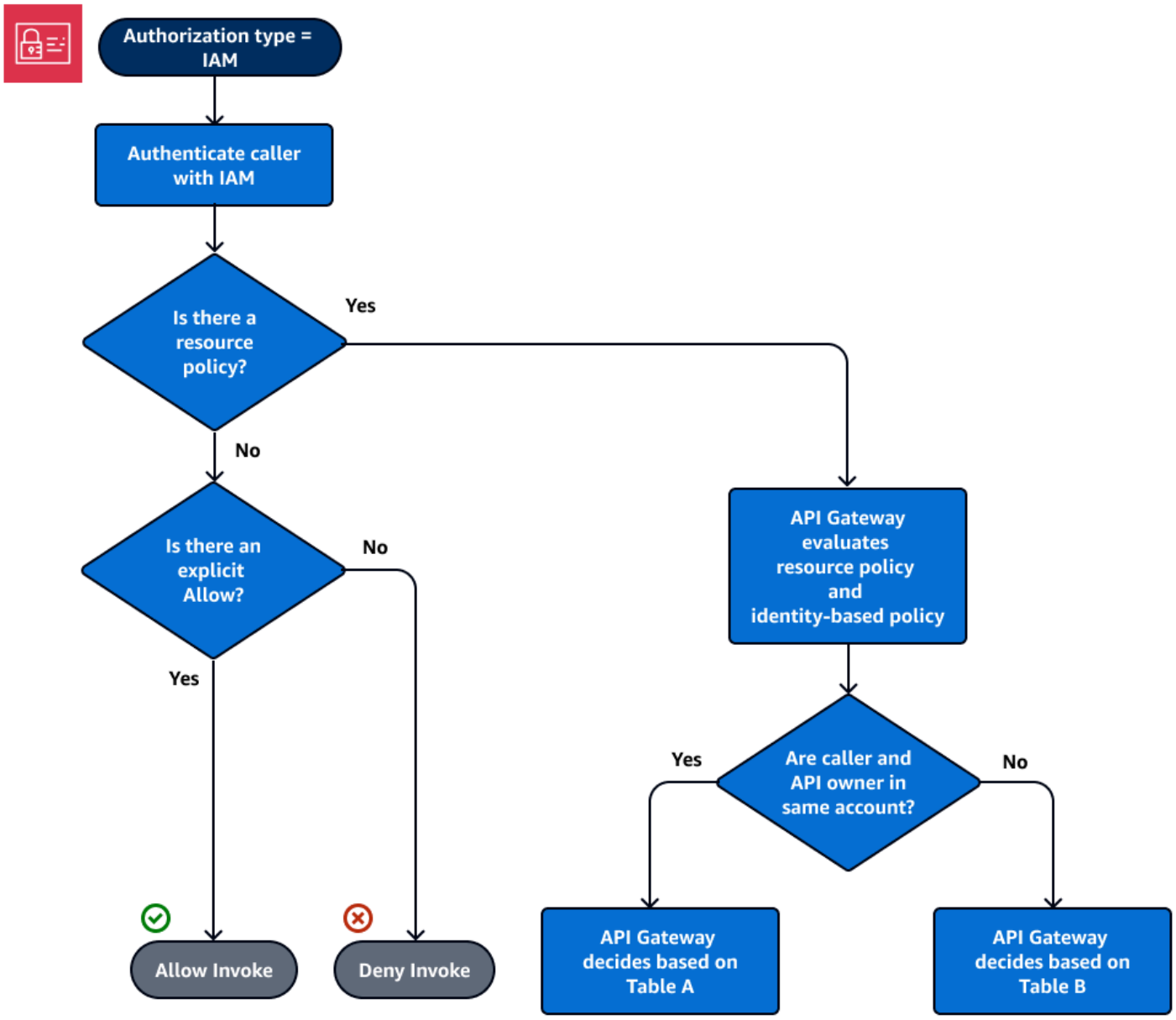
```
                "StringNotEquals": {
                    "aws:SourceVpce": "vpce-1a2b3c4d"
                }
            }
        }
    ]
}
```

Autenticación de IAM y política de recursos

En este flujo de trabajo, puede configurar la autenticación de IAM para la API, además de una política de recursos. Después de autenticar al usuario con el servicio de IAM, la API evalúa las políticas asociadas al usuario y a la política de recursos. El resultado varía en función de si el intermediario se encuentra en la misma Cuenta de AWS o una Cuenta de AWS diferente, del propietario de la API.

Si el intermediario y el propietario de la API proceden de cuentas distintas, las políticas de IAM y la política de recursos permiten explícitamente al intermediario continuar. Para obtener más información, consulte la Tabla B.

Sin embargo, si el intermediario y el propietario de la API se encuentran en la misma Cuenta de AWS, las políticas del usuario de IAM o la política de recursos deben permitir explícitamente al intermediario continuar. Para obtener más información, consulte la Tabla A.



El siguiente ejemplo muestra una política de recursos entre cuentas. Si presuponemos que la política de IAM contiene un efecto Allow (Permitir), esta política de recursos solo permite llamadas desde la VPC cuyo ID de VPC sea *vpc-2f09a348*. Para obtener más información, consulte la Tabla B.

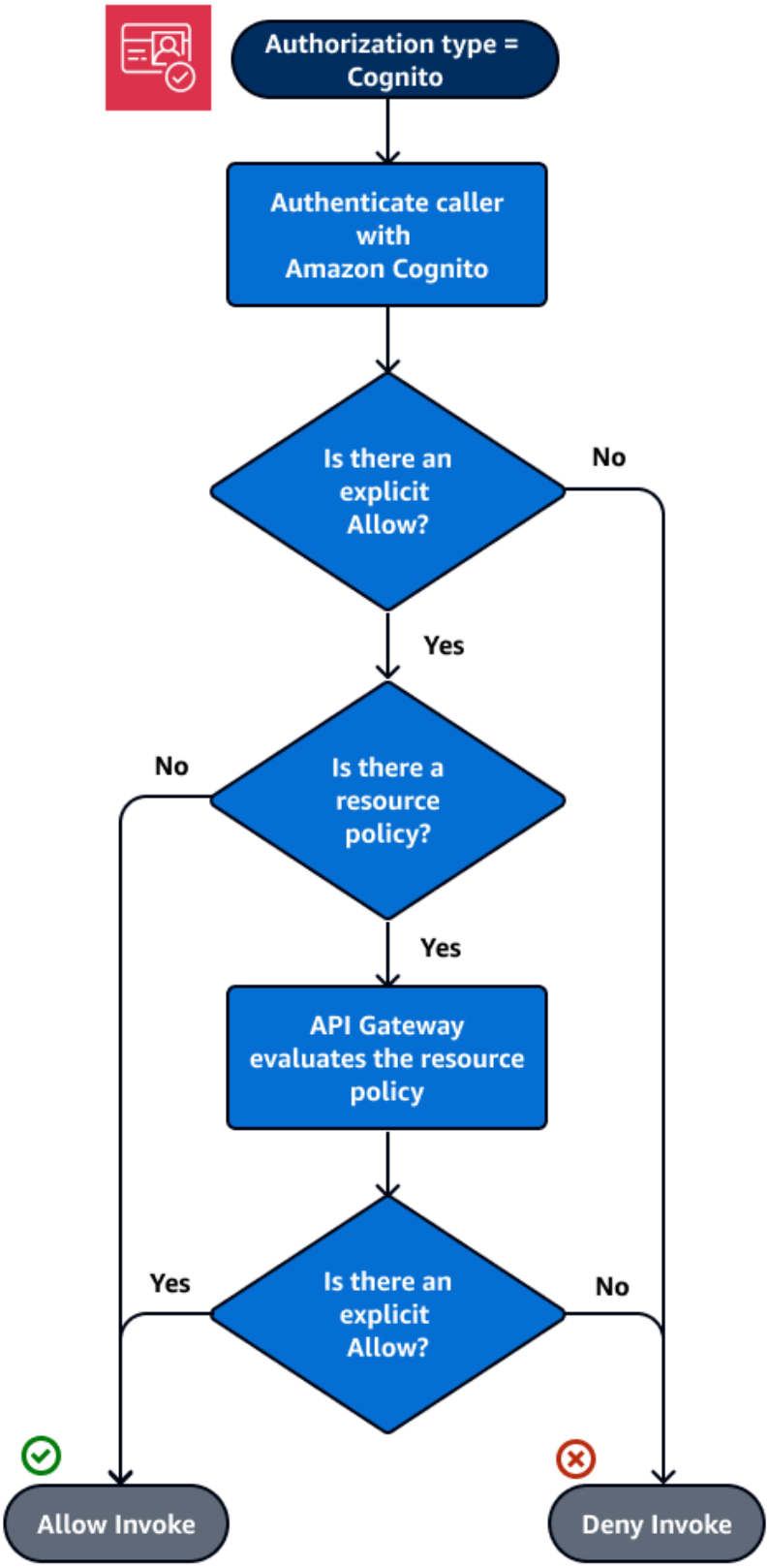
JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "arn:aws:execute-api:us-east-1:111111111111:api-id/"
            ],
            "Condition" : {
                "StringEquals": {
                    "aws:SourceVpc": "vpc-2f09a348"
                    }
            }
        }
    ]
}
```

Política de recursos y autenticación de Amazon Cognito

En este flujo de trabajo, se configura un grupo de usuarios de Amazon Cognito para la API, además de una política de recursos. API Gateway intenta primero autenticar a la persona que llama a través de Amazon Cognito. Esto se realiza normalmente a través de un token JWT proporcionado por la persona que llama. Si la autenticación se realiza correctamente, la política de recursos se evalúa de forma independiente, y se requiere un permiso explícito. Una denegación o "ni permitir ni denegar" da como resultado una denegación. El siguiente ejemplo muestra una política de recursos que podrían utilizarse junto con grupos de usuarios de Amazon Cognito.


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

El siguiente ejemplo muestra una política de recursos que solo permite llamadas desde direcciones IP de origen específicas, suponiendo que el token de autenticación de Amazon Cognito contiene un permiso. Para obtener más información, consulte la Tabla B.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "arn:aws:execute-api:us-east-1:111111111111:api-id/",
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": ["192.0.2.0/24", "198.51.100.0/24" ]
                }
            }
        }
    ]
}
```

Tablas de resultados de evaluación de políticas

La tabla A muestra el comportamiento resultante cuando el acceso a una API de API Gateway está controlado por una política de IAM o un autorizador de Lambda y una política de recursos de API Gateway, que se encuentran en la misma Cuenta de AWS.

Política de IAM (o autorizador de Lambda)	Política de recursos de API Gateway	Comportamiento resultante
Allow	Allow	Allow
Allow	Ni permitir ni denegar	Allow
Allow	Denegar	Denegación explícita
Ni permitir ni denegar	Allow	Allow

Política de IAM (o autorizador de Lambda)	Política de recursos de API Gateway	Comportamiento resultante
Ni permitir ni denegar	Ni permitir ni denegar	Denegar de forma implícita
Ni permitir ni denegar	Denegar	Denegación explícita
Denegar	Allow	Denegación explícita
Denegar	Ni permitir ni denegar	Denegación explícita
Denegar	Denegar	Denegación explícita

La tabla B muestra el comportamiento resultante cuando el acceso a una API de API Gateway está controlado por una política de IAM o un autorizador de grupos de usuarios de Amazon Cognito y una política de recursos de API Gateway, que se encuentran en diferentes Cuentas de AWS. Si no existe un permiso ni una denegación, el acceso entre cuentas se deniega. Esto se debe a que el acceso entre cuentas requiere que la política de recursos y la política de IAM o el autorizador de grupos de usuarios de Amazon Cognito concedan acceso de forma explícita.

Política de IAM (o autorizador de grupos de usuarios de Amazon Cognito)	Política de recursos de API Gateway	Comportamiento resultante
Allow	Allow	Allow
Allow	Ni permitir ni denegar	Denegar de forma implícita
Allow	Denegar	Denegación explícita
Ni permitir ni denegar	Allow	Denegar de forma implícita
Ni permitir ni denegar	Ni permitir ni denegar	Denegar de forma implícita
Ni permitir ni denegar	Denegar	Denegación explícita
Denegar	Allow	Denegación explícita
Denegar	Ni permitir ni denegar	Denegación explícita

Política de IAM (o autorizador de grupos de usuarios de Amazon Cognito)	Política de recursos de API Gateway	Comportamiento resultante
Denegar	Denegar	Denegación explícita

Ejemplos de políticas de recursos de API Gateway

En esta página se presentan algunos ejemplos de casos de uso típicos de políticas de recursos de API Gateway.

Las políticas de ejemplo siguientes utilizan una sintaxis simplificada para especificar el recurso de API. Esta sintaxis simplificada es una forma abreviada de hacer referencia a un recurso de API, en lugar de especificar el nombre de recursos de Amazon (ARN) completo. API Gateway convierte la sintaxis abreviada en el ARN completo al guardar la política. Por ejemplo, puede especificar el recurso `execute-api:/stage-name/GET/pets` en una política de recursos. API Gateway convierte el recurso en `arn:aws:execute-api:us-east-2:123456789012:aabbccddee/stage-name/GET/pets` cuando se guarda la política de recursos. API Gateway crea el ARN completo utilizando la región actual, el ID de su cuenta de AWS y el ID de la API REST con la que está asociada la política de recursos. Puede utilizar `execute-api:/*` para representar todas las etapas, métodos y rutas de la API actual. Para obtener información acerca del lenguaje de la política de acceso, consulte Información general del lenguaje de políticas de acceso para Amazon API Gateway.

Temas

- Ejemplo: permitir que los roles de otra cuenta de AWS utilicen una API
- Ejemplo: Cómo denegar el tráfico a una API para una dirección o un rango de direcciones IP de origen
- Ejemplo: Denegar el tráfico de API basado en la dirección IP de origen o rango cuando se utiliza una API privada
- Ejemplo: Permitir el tráfico de una API privada en función del punto de conexión de VPC o la VPC de origen

Ejemplo: permitir que los roles de otra cuenta de AWS utilicen una API

En la siguiente política de recursos de ejemplo, se otorga acceso a la API de una cuenta de AWS a dos roles de una cuenta de AWS diferente a través de los protocolos Signature Version 4 (SigV4) o Signature Version 4a (SigV4a). En concreto, se concede al rol de desarrollador y administrador de la cuenta de AWS identificados por *account-id-2* la acción `execute-api:Invoke` para ejecutar la acción `GET` en el recurso `pets` (API) en la cuenta de AWS.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:role/developer",
                    "arn:aws:iam::111122223333:role/Admin"
                ]
            },
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/stage/GET/pets"
            ]
        }
    ]
}
```

Ejemplo: Cómo denegar el tráfico a una API para una dirección o un rango de direcciones IP de origen

El siguiente ejemplo de política de recursos deniega (bloquea) el tráfico entrante a una API privada procedente de dos bloques de direcciones IP de origen especificadas.

JSON

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
               "execute-api:/*"
            ],
            "Condition" : {
                "IpAddress": {
                    "aws:SourceIp": ["192.0.2.0/24", "198.51.100.0/24" ]
                }
            }
        }
    ]
}
```

Si utiliza políticas de usuario de IAM o políticas de recursos de API Gateway para controlar el acceso a API Gateway o a cualquier API de API Gateway, confirme que las políticas estén actualizadas para incluir los rangos de direcciones IPv6. Las políticas que no se actualizan para gestionar las direcciones IPv6 pueden afectar al acceso del cliente a API Gateway cuando comiencen a utilizar el punto de conexión de pila doble. Para obtener más información, consulte the section called “Uso de direcciones IPv6 en políticas de IAM”.

Ejemplo: Denegar el tráfico de API basado en la dirección IP de origen o rango cuando se utiliza una API privada

El siguiente ejemplo de política de recursos deniega (bloquea) el tráfico entrante a una API privada procedente de dos bloques de direcciones IP de origen especificadas. Cuando se utilizan API privadas, el punto de conexión final de VPC para `execute-api` vuelve a escribir la dirección IP de origen original. La condición `aws:VpcSourceIp` filtra la solicitud contra la dirección IP del solicitante original.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
               "execute-api:/*"
            ],
            "Condition" : {
                "IpAddress": {
                    "aws:VpcSourceIp": ["192.0.2.0/24", "198.51.100.0/24"]
                }
            }
        }
    ]
}
```

Ejemplo: Permitir el tráfico de una API privada en función del punto de conexión de VPC o la VPC de origen

En el ejemplo siguiente, las políticas de recursos permiten el tráfico entrante en una API privada solo desde una nube virtual privada (VPC) o un punto de conexión de VPC especificados.

Esta política de recursos de ejemplo especifica una VPC de origen:

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ],
            "Condition" : {
                "StringNotEquals": {
                   "aws:SourceVpc": "vpc-1a2b3c4d"
                }
            }
        }
    ]
}
```

Esta política de recursos de ejemplo especifica un punto de conexión de VPC de origen:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        {
```

```
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ],
            "Condition" : {
                "StringNotEquals": {
                    "aws:SourceVpce": "vpce-1a2b3c4d"
                }
            }
        }
    ]
}
```

Creación y asociación de una política de recursos de API Gateway a una API

Para permitir a un usuario acceder a la API llamando al servicio de ejecución de la API, debe crear una política de recursos de API Gateway y asociar la política a la API. Al asociar una política a la API, se aplican los permisos de la política a los métodos de la API. Si actualiza la política de recursos, tendrá que implementar la API.

Temas

- Requisitos previos
- Asociación de una política de recursos a una API de API Gateway
- Solución de problemas de política de recursos

Requisitos previos

Para actualizar una política de recursos de API Gateway, necesitará el permiso `apigateway:UpdateRestApiPolicy` y el permiso `apigateway:PATCH`.

Para una API regional u optimizada para sistemas perimetrales, puede adjuntar la política de recursos a la API a medida que la crea o después de implementarla. Para una API privada, no puede implementar la API sin una política de recursos. Para obtener más información, consulte the section called “API de REST privadas”.

Asociación de una política de recursos a una API de API Gateway

El siguiente procedimiento muestra cómo asociar una política de recursos a una API de API Gateway.

Consola de administración de AWS

Para asociar una política de recursos a una API de API Gateway

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST.
3. En el panel de navegación principal, elija Política de recursos.
4. Elija Crear política.
5. (Opcional) Elija Seleccionar una plantilla para generar una política de ejemplo.

   En las políticas de ejemplo, los marcadores de posición se indican mediante llaves ("{{*placeholder*}}"). Sustituya cada uno de los marcadores de posición, incluidas las llaves, por la información necesaria.
6. Si no utiliza uno de los ejemplos de plantilla, ingrese la política de recursos.
7. Seleccione Save changes (Guardar cambios).

Si la API se ha implementado anteriormente en la consola de API Gateway, tendrá que volver a implementarla para que la política de recursos surta efecto.

AWS CLI

Para usar la AWS CLI para crear una nueva API y asociarla a una política de recursos, use el siguiente comando create-rest-api:

```
aws apigateway create-rest-api \
    --name "api-name" \
    --policy "{\"jsonEscapedPolicyDocument\"}"
```

Para usar la AWS CLI para asociar una política de recursos a una API existente, use el siguiente comando update-rest-api:

```
aws apigateway update-rest-api \
    --rest-api-id api-id \
```

```
    --patch-operations op=replace,path=/
policy,value='"{\"jsonEscapedPolicyDocument\"}"'
```

También puede adjuntar la política de recursos como archivo `policy.json` independiente e incluirla en el comando create-rest-api. El siguiente comando create-rest-api crea una nueva API con una política de recursos:

```
aws apigateway create-rest-api \
    --name "api-name" \
    --policy file://policy.json
```

El archivo `policy.json` es una política de recursos de API Gateway, como the section called "Ejemplo: Cómo denegar el tráfico a una API para una dirección o un rango de direcciones IP de origen".

AWS CloudFormation

Puede utilizar CloudFormation para crear una API con una política de recursos. En el siguiente ejemplo, se crea una API de REST con la política de recursos de ejemplo, the section called "Ejemplo: Cómo denegar el tráfico a una API para una dirección o un rango de direcciones IP de origen".

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Name: testapi
      Policy:
        Statement:
          - Action: 'execute-api:Invoke'
            Effect: Allow
            Principal: '*'
            Resource: 'execute-api:/*'
          - Action: 'execute-api:Invoke'
            Effect: Deny
            Principal: '*'
            Resource: 'execute-api:/*'
            Condition:
              IpAddress:
                'aws:SourceIp': ["192.0.2.0/24", "198.51.100.0/24" ]
        Version: 2012-10-17
```

```
  Resource:
    Type: 'AWS::ApiGateway::Resource'
    Properties:
      RestApiId: !Ref Api
      ParentId: !GetAtt Api.RootResourceId
      PathPart: 'helloworld'
  MethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref Resource
      HttpMethod: GET
      ApiKeyRequired: false
      AuthorizationType: NONE
      Integration:
        Type: MOCK
        RequestTemplates:
          application/json: '{"statusCode": 200}'
        IntegrationResponses:
          - StatusCode: 200
            ResponseTemplates:
              application/json: '{}'
      MethodResponses:
        - StatusCode: 200
          ResponseModels:
            application/json: 'Empty'
  ApiDeployment:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn:
      - MethodGet
    Properties:
      RestApiId: !Ref Api
      StageName: test
```

Solución de problemas de política de recursos

La siguiente guía de solución de problemas puede ayudar a resolver los problemas relacionados con la política de recursos.

Mi API devuelve {"Mensaje": El usuario: anónimo no está autorizado a realizar: execute-api:Invoke en resource: arn:aws:execute-api:us-east-1:********/****/****/"}

En la política de recursos, si establece la entidad principal en una entidad principal de AWS, por ejemplo:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111111111111:role/developer",
                    "arn:aws:iam::111111111111:role/Admin"
                ]
            },
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/stage/GET/pets"
            ]
        }
    ]
}
```

Debe usar la autorización de `AWS_IAM` para todos los métodos de la API o, de lo contrario, la API devolverá el mensaje de error anterior. Para obtener más instrucciones sobre cómo activar la autorización de `AWS_IAM` para un método, consulte the section called “Métodos”.

Mi política de recursos no se actualiza

Si actualiza la política de recursos después de crear la API, tendrá que implementar la API para propagar los cambios después de asociar la política actualizada. Si únicamente se actualiza o se guarda la política, no se modifica el comportamiento en tiempo de ejecución de la API. Para obtener más información acerca cómo implementar una API, consulte the section called “Implementación de las API de REST”.

Mi política de recursos devuelve el siguiente error: Documento de política no válido. Compruebe la sintaxis de la política y asegúrese de que las entidades principales sean válidas.

Para solucionar este error, primero le recomendamos que compruebe la sintaxis de la política. Para obtener más información, consulte the section called “Información general del lenguaje de políticas de acceso para Amazon API Gateway”. También le recomendamos que compruebe que todas las entidades principales especificadas sean válidas y que no se hayan eliminado.

Además, si la API se encuentra en una región de suscripción, compruebe que todas las cuentas de la política de recursos tengan la región habilitada.

AWSClaves de condición de que se pueden utilizar en las políticas de recursos de API Gateway

La siguiente tabla contiene las claves de condición de AWS que se pueden utilizar en las políticas de recursos de las API de API Gateway con cada tipo de autorización.

Para obtener más información acerca de las claves de condición de AWS, consulte Claves de contexto de condición global de AWS.

Claves de condición	Criterios	Necesita **AuthN**?	Tipo de autorización
`aws:CurrentTime`	Ninguno	No	Todos
`aws:EpochTime`	Ninguno	No	Todos
`aws:TokenIssueTime`	La clave solo está en las solicitudes que se firman con credenciales de seguridad temporales.	Sí	IAM
`aws:MultiFactorAuthPresent`	La clave solo está en las solicitudes que se firman con credenciales de seguridad temporales.	Sí	IAM
`aws:MultiFactorAuthAge`	La clave solo está si se utiliza la MFA en las solicitudes.	Sí	IAM

Claves de condición	Criterios	Necesita **AuthN**?	Tipo de autorización
`aws:PrincipalAccount`	Ninguno	Sí	IAM
`aws:PrincipalArn`	Ninguno	Sí	IAM
`aws:PrincipalOrgID`	Esta clave solamente se incluye en el contexto de la solicitud si la entidad principal es miembro de una organización.	Sí	IAM
`aws:PrincipalOrgPaths`	Esta clave solamente se incluye en el contexto de la solicitud si la entidad principal es miembro de una organización.	Sí	IAM
`aws:PrincipalTag`	Esta clave solamente se incluye en el contexto de la solicitud si la entidad principal está usando un usuario de IAM con etiquetas asociadas. Se incluye para una entidad principal que utiliza un rol de IAM con etiquetas o etiquetas de sesión asociadas.	Sí	IAM
`aws:PrincipalType`	Ninguno	Sí	IAM

Claves de condición	Criterios	Necesita **AuthN**?	Tipo de autorización
`aws:Referer`	La clave solo está si el valor lo proporciona el intermediario en el encabezado HTTP.	No	Todos
`aws:SecureTransport`	Ninguno	No	Todos
`aws:SourceArn`	Ninguno	No	Todos
`aws:SourceIp`	Ninguno	No	Todos
`aws:SourceVpc`	Esta clave solo se puede utilizar con las API privadas.	No	Todos
`aws:SourceVpce`	Esta clave solo se puede utilizar con las API privadas.	No	Todos
`aws:VpcSourceIp`	Esta clave solo se puede utilizar con las API privadas.	No	Todos
`aws:UserAgent`	La clave solo está si el valor lo proporciona el intermediario en el encabezado HTTP.	No	Todos
`aws:userid`	Ninguno	Sí	IAM
`aws:username`	Ninguno	Sí	IAM

Control del acceso a una API de REST con permisos de IAM

Puede controlar el acceso a su API de Amazon API Gateway con permisos de IAM controlando el acceso a los dos procesos de componentes de API Gateway siguientes:

- Para crear, implementar y administrar una API en API Gateway, debe conceder al desarrollador de la API permisos para realizar las acciones necesarias admitidas por el componente de administración de la API de API Gateway.
- Para llamar a una API implementada o para actualizar el almacenamiento en caché de la API, debe conceder al intermediario de la API permisos para realizar las acciones de IAM necesarias admitidas por el componente de ejecución de la API de API Gateway.

El control de acceso para los dos procesos implica diferentes modelos de permisos, que se explican a continuación.

Modelo de permisos de API Gateway para crear y administrar una API

Para permitir a un desarrollador de API crear y administrar una API en API Gateway debe crear políticas de permisos de IAM que permitan a un desarrollador de la API especificado crear, actualizar, implementar, ver o eliminar las entidades de API necesarias. Al asociar la política de permisos a un usuario, rol o grupo.

Para dar acceso, agregue permisos a los usuarios, grupos o roles:

- Usuarios y grupos en AWS IAM Identity Center:

  Cree un conjunto de permisos. Siga las instrucciones de Creación de un conjunto de permisos en la Guía del usuario de AWS IAM Identity Center.
- Usuarios gestionados en IAM a través de un proveedor de identidades:

  Cree un rol para la federación de identidades. Siga las instrucciones descritas en Creación de un rol para un proveedor de identidad de terceros (federación) en la Guía del usuario de IAM.
- Usuarios de IAM:
  - Cree un rol que el usuario pueda aceptar. Siga las instrucciones descritas en Creación de un rol para un usuario de IAM en la Guía del usuario de IAM.
  - (No recomendado) Adjunte una política directamente a un usuario o agregue un usuario a un grupo de usuarios. Siga las instrucciones descritas en Adición de permisos a un usuario (consola) de la Guía del usuario de IAM.

Para obtener más información sobre cómo usar este modelo de permisos, consulte the section called “Políticas basadas en identidades de API Gateway”.

Modelo de permisos de API Gateway para invocar una API

Para permitir a un intermediario de la API invocar la API o actualizar su caché, debe crear políticas de IAM que permitan a un intermediario de la API especificado invocar el método de API para el que se ha habilitado la autenticación de usuarios. El desarrollador de la API establece la propiedad `authorizationType` del método en `AWS_IAM` para exigir que el intermediario envíe las credenciales del usuario que se va a autenticar. API Gateway admite Signature Version 4a (SigV4a) y Signature Version 4 (SigV4) para autenticar las credenciales del usuario. Para obtener más información, consulte AWS Signature Version 4. A continuación, adjunte la política a un usuario, rol o grupo.

En esta instrucción de política de permisos de IAM, el elemento `Resource` de IAM contiene una lista de métodos de la API implementada identificados por verbos HTTP y rutas de recursos de API Gateway. El elemento `Action` de IAM contiene las acciones de ejecución de API de API Gateway necesarias. Estas acciones incluyen `execute-api:Invoke` o `execute-api:InvalidateCache`, donde `execute-api` designa el componente de ejecución de la API subyacente de API Gateway.

Para obtener más información sobre cómo usar este modelo de permisos, consulte Controlar el acceso para invocar una API.

Cuando una API está integrada con un servicio de AWS (por ejemplo, AWS Lambda) en el backend, API Gateway también debe tener permisos para obtener acceso a los recursos de AWS integrados (por ejemplo, invocar una función de Lambda) en nombre de la persona que llama a la API. Para otorgar estos permisos, cree un rol de IAM del tipo servicio de AWS para API Gateway. Cuando crea este rol en la consola de administración de IAM, este rol resultante contiene la siguiente política de confianza de IAM que designa a API Gateway como la entidad de confianza capaz de asumir el rol:

JSON

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

```
    }
  ]
}
```

Si crea el rol de IAM llamando al comando create-role de la CLI o a un método del SDK correspondiente, debe proporcionar la política de confianza anterior como el parámetro de entrada de `assume-role-policy-document`. No intente crear esta política directamente en la consola de administración de IAM ni llamando al comando de la AWS CLI create-policy ni a un método del SDK correspondiente.

Para que API Gateway llame al servicio de AWS integrado, también debe adjuntar a este rol las políticas de permisos de IAM correspondientes para llamar a los servicios de AWS integrados. Por ejemplo, para llamar a una función de Lambda, debe incluir la siguiente política de permisos de IAM en el rol de IAM:

JSON

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

Tenga en cuenta que Lambda admite la política de acceso basada en recursos, que combina políticas de confianza y de permisos. Cuando integre una API con una función de Lambda mediante la consola de API Gateway, no se le pedirá que establezca este rol de IAM explícitamente, porque la consola establece los permisos basados en recursos en la función de Lambda por usted, con su consentimiento.

> **Note**
>
> Para establecer el control del acceso a un servicio de AWS, puede utilizar el modelo de permisos basado en el intermediario, donde una política de permisos se asocia directamente

al usuario o grupo del intermediario, o el modelo de permisos basado en rol, donde una política de permisos se asocia a un rol de IAM que API Gateway puede asumir. Los permisos de políticas pueden ser diferentes en los dos modelos. Por ejemplo, la política basada en el intermediario bloquea el acceso, mientras que la política basada en roles lo permite. Puede aprovechar esto para exigir que un usuario tenga acceso a un servicio de AWS solo a través de la API de API Gateway.

Controlar el acceso para invocar una API

En esta sección, obtendrá información sobre el modelo de permisos para controlar el acceso a su API mediante permisos de IAM. Cuando la autorización de IAM está habilitada, los clientes deben utilizar Signature Version 4a (SigV4a) y Signature Version 4 (SigV4) para firmar las solicitudes con credenciales de AWS. Para obtener más información, consulte AWS Signature Version 4.

En esta sección, mostramos una plantilla de declaración de política de IAM y la referencia de la declaración de política. La referencia de la declaración de política incluye los formatos de los campos `Action` y `Resource` relacionados con el servicio de ejecución de la API. Utilice estas referencias para crear su declaración de política de IAM. Cuando cree la declaración de política de IAM, es posible que deba tener en cuenta la forma en que las políticas de recursos de API Gateway afectan al flujo de trabajo de autorización. Para obtener más información, consulte the section called “Cómo afectan las políticas de recursos al flujo de trabajo de autorización”.

Para API privadas, debe utilizar una combinación de una política de recursos de API Gateway y una política de punto de enlace de la VPC. Para obtener más información, consulte los siguientes temas:

- the section called “Uso de políticas de recursos de API Gateway”
- the section called “Uso de políticas de punto de conexión de VPC para API privadas”

Controlar quién puede llamar a una API de API Gateway con políticas de IAM

Para controlar quién puede o no puede llamar a una API implementada con permisos de IAM, cree un documento de política de IAM con los permisos necesarios. A continuación, se muestra una plantilla para un documento de política como este.

JSON

```
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Permission",
      "Action": [
        "execute-api:Execution-operation"
      ],
      "Resource": [
        "arn:aws:execute-api:region:123456789012:api-
id/stage/METHOD_HTTP_VERB/Resource-path"
      ]
    }
  ]
}
```

Aquí, *Permission* se sustituirá por `Allow` o `Deny` dependiendo de si desea conceder o revocar los permisos incluidos. *Execution-operation* se sustituirá por las operaciones compatibles con el servicio de ejecución de la API. *METHOD_HTTP_VERB* hace referencia a un verbo HTTP compatible con los recursos especificados. *Resource-path* es el marcador de posición de la ruta URL de una instancia de `Resource` de la API implementada que admite el *METHOD_HTTP_VERB* mencionado. Para obtener más información, consulte Referencia de instrucciones de políticas de IAM para ejecutar la API en API Gateway.

> **Note**
>
> Para que las políticas de IAM sean eficaces, debe haber habilitado la autenticación de IAM en los métodos de API configurando `AWS_IAM` para la propiedad `authorizationType` de los métodos. En caso contrario, estos métodos de API serán accesibles públicamente.

Por ejemplo, para conceder a un usuario permiso para ver una lista de mascotas expuesta por una API determinada, pero denegarle permiso para agregar una mascota a la lista, se podría incluir la siguiente instrucción en la política de IAM:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:us-east-1:111111111111:api-id/*/GET/pets"
      ]
    },
    {
      "Effect": "Deny",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:us-east-1:111111111111:api-id/*/POST/pets"
      ]
    }
  ]
}
```

Para conceder a un usuario permiso para ver una mascota específica expuesta por una API que se configura como `GET /pets/{petId}`, podría incluir la siguiente declaración en la política de IAM:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "execute-api:Invoke"
            ],
            "Resource": [
                "arn:aws:execute-api:us-east-1:111122223333:api-id/*/
GET/pets/a1b2"
            ]
        }
    ]
}
```

Referencia de instrucciones de políticas de IAM para ejecutar la API en API Gateway

La siguiente información describe el formato de Action y Resource de las instrucciones de política de IAM de permisos de acceso para ejecutar una API.

Formato de Action de permisos para ejecutar la API en API Gateway

La expresión `Action` de ejecución de API tiene el siguiente formato general:

```
execute-api:action
```

donde *`action`* es una acción de ejecución de API disponible:

- *, que representa todas las acciones siguientes.
- Invoke, que se utiliza para invocar una API previa solicitud del cliente.
- InvalidateCache, que se utiliza para invalidar la caché de API previa solicitud del cliente.

Formato de Resource de permisos para ejecutar la API en API Gateway

La expresión `Resource` de ejecución de API tiene el siguiente formato general:

```
arn:aws:execute-api:region:account-id:api-id/stage-name/HTTP-VERB/resource-path-specifier
```

donde:

- *`region`* es la región de AWS (como **`us-east-1`** o **`*`** para todas las regiones de AWS) que se corresponde con la API implementada para el método.
- *`account-id`* es el ID de cuenta de 12 dígitos de AWS del propietario de la API de REST.
- *`api-id`* es el identificador que API Gateway asignó a la API para el método.
- *`stage-name`* es el nombre de la etapa asociada al método.
- *`HTTP-VERB`* es el verbo HTTP del método. Puede ser uno de las siguientes: GET, POST, PUT, DELETE, PATCH.
- *`resource-path-specifier`* es la ruta al método deseado.

> **Note**
>
> Si especifica un comodín (*), la expresión `Resource` aplica el comodín al resto de la expresión.

Algunos ejemplos de expresiones de recursos incluyen:

- **`arn:aws:execute-api:*:*:*`** para cualquier ruta de recurso en cualquier etapa, para cualquier API de cualquier región de AWS.
- **`arn:aws:execute-api:us-east-1:*:*`** para cualquier ruta de recurso en cualquier etapa, para cualquier API en la región de AWS `us-east-1`.
- **`arn:aws:execute-api:us-east-1:*:`*`api-id`*`/*`** para cualquier ruta de recurso en cualquier etapa, para la API con el identificador *`api-id`* de la región de AWS us-east-1.
- **`arn:aws:execute-api:us-east-1:*:`*`api-id`*`/test/*`** para cualquier ruta de recurso en la etapa de `test`, para la API con el identificador de *`api-id`* de la región de AWS us-east-1.

Para obtener más información, consulte Referencia del nombre de recurso de Amazon (ARN) de API Gateway.

Ejemplos de política de IAM para permisos de ejecución de la API

Para ver el modelo de permisos y otra información de referencia, consulte Controlar el acceso para invocar una API.

La siguiente instrucción de política otorga al usuario permiso para llamar a cualquier método POST en la ruta `mydemoresource`, en la etapa `test`, para la API con el identificador `a123456789`, suponiendo que la API correspondiente se ha implementado en la región de AWS us-east-1:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
```

```
      ],
      "Resource": [
        "arn:aws:execute-api:us-east-1:*:a123456789/test/POST/my-demo-resource-
path/*"
      ]
    }
  ]
}
```

La siguiente instrucción de política de ejemplo concede al usuario permiso para llamar a cualquier método en la ruta del recurso `petstorewalkthrough/pets`, en cualquier etapa, para la API con el identificador `a123456789`, en cualquier región de AWS en la que se haya implementado la API correspondiente:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:*:*:a123456789/*/*/petstorewalkthrough/pets"
      ]
    }
  ]
}
```

Uso de políticas de punto de conexión de VPC para API privadas en API Gateway

Para mejorar la seguridad de la API privada, puede crear una política de punto de conexión de VPC. Una política de punto de conexión de VPC es una política de recursos de IAM que se adjunta a un punto de conexión de VPC. Para obtener más información, consulte Control del acceso a los servicios con Puntos de conexión de VPC.

Es posible que desee crear una política de punto de conexión de VPC para realizar las siguientes tareas.

- Permita que solo determinadas organizaciones o recursos accedan al punto de conexión de VPC e invoquen la API.
- Use una sola política y evite las políticas basadas en sesiones o roles para controlar el tráfico a la API.
- Refuerce el perímetro de seguridad de la aplicación al migrar de en las instalaciones a AWS.

Consideraciones de la política del punto de conexión de VPC

A continuación, se incluyen algunas consideraciones para la política de puntos de conexión de VPC:

- La identidad del invocador se evalúa en función del valor `Authorization` del encabezado. Primero se evalúa la política de puntos de conexión de VPC y, a continuación, API Gateway evalúa la solicitud en función del tipo de autorización configurado en la solicitud del método. En la siguiente tabla se muestra cómo se evalúa la política de puntos de conexión de VPC en función del contenido del valor del encabezado `Authorization`.

Contenido del valor del encabezado **`Authorization`**	Cómo se evalúa la política de punto de conexión de VPC
Sin contenido	El invocador se evalúa como un usuario anónimo
Firma SigV4 o SigV4a válida	El invocador se evalúa como la identidad de IAM autenticada a partir de la firma
Firma SigV4 o SigV4a no válida	API Gateway deniega el acceso
Información de autorización que no es SigV4, como un token de portador	El invocador se evalúa como un usuario anónimo

- Si el control de acceso depende del uso de un token de portador, como un autorizador de Lambda o Amazon Cognito, puede controlar el perímetro de seguridad mediante las propiedades del recurso.
- Si los controles de autorización utilizan la autorización de IAM, puede controlar el perímetro de seguridad mediante las propiedades del recurso y las propiedades de la entidad principal.

- Las políticas de punto de conexión de VPC se pueden utilizar junto con políticas de recursos de API Gateway. La política de recursos de API Gateway especifica qué entidades principales pueden acceder a la API. La política de puntos de conexión especifica quién puede acceder a la VPC y qué API se puede llamar desde el punto de conexión de VPC. La API privada necesita una política de recursos pero no necesita crear una política de punto de conexión de VPC personalizada.

Ejemplos de políticas de puntos de conexión de VPC

Puede crear políticas para los puntos de conexión de Amazon Virtual Private Cloud para Amazon API Gateway en las que puede especificar lo siguiente.

- La entidad principal que puede realizar acciones.
- Las acciones que se pueden realizar.
- Los recursos en los que se pueden realizar acciones.

Esto puede depender del contenido del encabezado de autorización. Para obtener más información, consulte the section called “Consideraciones de la política del punto de conexión de VPC”. Para obtener ejemplos de políticas adicionales, consulte Data perimeter policy examples en el sitio web de GitHub.

Para asociar la política al punto de conexión de VPC, tendrá que utilizar la consola de VPC. Para obtener más información, consulte Control del acceso a los servicios con Puntos de conexión de VPC.

Ejemplo 1: Política de punto de conexión de VPC que otorga acceso a dos API

El siguiente ejemplo de política otorga acceso solo a dos API específicas a través del punto de conexión de VPC al que está asociado la política.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Principal": "*",
            "Action": [
                "execute-api:Invoke"
```

```
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:execute-api:us-east-1:123412341234:a1b2c3d4e5/*",
                "arn:aws:execute-api:us-east-1:123412341234:aaaaa11111/*"
            ]
        }
    ]
}
```

Ejemplo 2: Política de punto de conexión de VPC que otorga acceso a métodos GET

El siguiente ejemplo de política otorga acceso a los usuarios a los métodos `GET` para una API específica a través del punto de conexión de VPC al que está asociada la política.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Principal": "*",
            "Action": [
                "execute-api:Invoke"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:execute-api:us-east-1:123412341234:a1b2c3d4e5/stageName/
GET/*"
            ]
        }
    ]
}
```

Ejemplo 3: Política de punto de conexión de VPC que otorga acceso a un usuario específico a una API específica

El siguiente ejemplo de política otorga acceso a un usuario específico a una API específica a través del punto de conexión de VPC al que está asociada la política.

En este caso, como la política restringe el acceso a entidades principales de IAM específicas, debe establecer el `authorizationType` del método en `AWS_IAM` o `NONE`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Principal": {
                "AWS": [
                    "arn:aws:iam::123412341234:user/MyUser"
                ]
            },
            "Action": [
                "execute-api:Invoke"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:execute-api:us-east-1:123412341234:a1b2c3d4e5/*"
            ]
        }
    ]
}
```

Ejemplo 4: Política de punto de conexión de VPC que otorga a los usuarios acceso a un nombre de dominio personalizado específico y a todas las API asignadas al dominio

El siguiente ejemplo de política otorga acceso a los usuarios a un determinado nombre de dominio personalizado para API privadas a través del punto de conexión de VPC al que está asociada la política. Con esta política, siempre que el usuario haya creado una asociación de acceso a nombre de dominio entre el punto de conexión de VPC y el nombre de dominio personalizado y se le otorgue acceso para invocar el nombre de dominio personalizado y cualquier API privada asignada al nombre de dominio personalizado, el usuario puede invocar cualquier API asignada a este nombre de dominio personalizado. Para obtener más información, consulte the section called “Nombres de dominio personalizados para API privadas”.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "execute-api:Invoke",
      "Resource": [
        "*"
      ],
       "Condition": {
        "ArnEquals": {
          "execute-api:viaDomainArn": "arn:aws:execute-api:us-
west-2:111122223333:/domainnames/private.test.com+f4g5h6"
        }
      }
    }
  ]
}
```

Ejemplo 5: Política de punto de conexión de VPC que otorga o deniega el acceso a determinadas API y recursos de dominio

La siguiente política de ejemplo otorga a los usuarios acceso a determinadas API y recursos de dominio. Con esta política, siempre que el usuario haya creado una asociación de acceso a nombre de dominio entre el punto de conexión de VPC y el nombre de dominio personalizado y se le otorgue acceso para invocar el nombre de dominio personalizado y cualquier API privada asignada al nombre de dominio personalizado, el usuario puede invocar las API privadas y los recursos de dominio permitidos. Para obtener más información, consulte the section called “Nombres de dominio personalizados para API privadas”.

JSON

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "execute-api:Invoke",
      "Resource": [
        "arn:aws:execute-api:us-west-2:111122223333:/domainnames/private.test.com
+f4g5h6",
        "arn:aws:execute-api:us-west-2:111122223333:a1b2c3d4e5/*"
      ]
    },
    {
      "Effect": "Deny",
      "Principal": {
        "AWS": "*"
      },
      "Action": "execute-api:Invoke",
      "Resource": [
        "arn:aws:execute-api:us-west-2:111122223333:a1b2c3d4e5/admin/*",
        "arn:aws:execute-api:us-west-2:111122223333:bcd123455/*"
      ]
    }
  ]
}
```

Ejemplo 6: política de punto de conexión de VPC que concede o deniega el acceso a entidades principales y recursos que pertenecen a una organización

La siguiente política de ejemplo concede acceso a entidades principales y recursos que pertenecen a una organización.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Condition": {
                "StringEquals": {
```

```
                    "aws:ResourceOrgID": "o-abcd1234",
                    "aws:PrincipalOrgID": "o-abcd1234"
                }
            },
            "Action": "*",
            "Resource": "*",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Sid": "AllowRequestsByOrgsIdentitiesToOrgsResources"
        }
    ]
}
```

Uso de etiquetas para controlar el acceso a las API de REST en API Gateway

El permiso para obtener acceso a las API REST se puede ajustar mediante el control de acceso basado en atributos en las políticas de IAM.

Para obtener más información, consulte the section called “Control de acceso basado en atributos”.

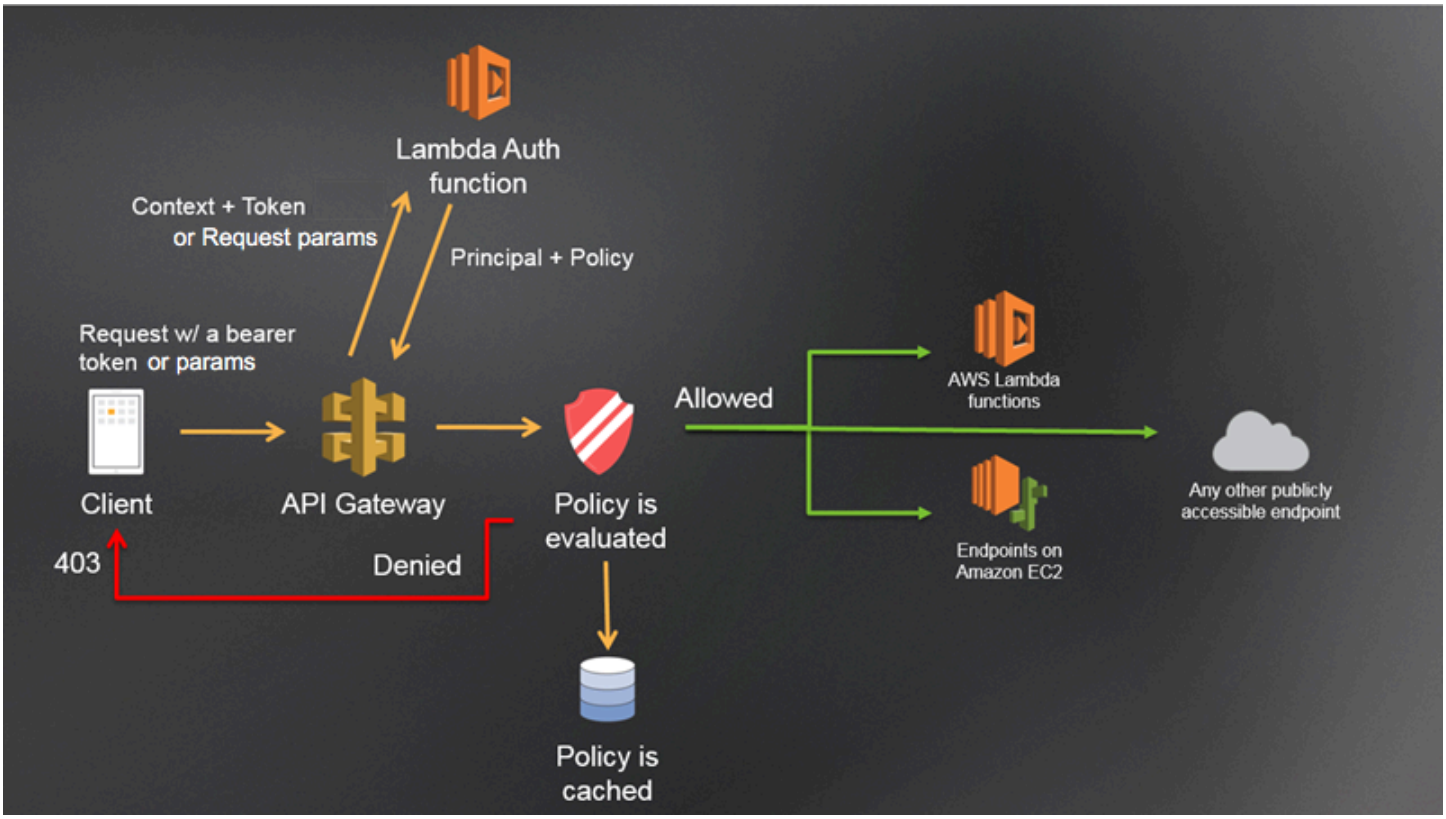
Uso de autorizadores Lambda de API Gateway

Utilice un autorizador de Lambda (que anteriormente se denominaba autorizador personalizado) para controlar el acceso a su API. Cuando un cliente realiza una solicitud al método de la API, API Gateway llama al autorizador de Lambda. El autorizador de Lambda toma la identidad del intermediario como entrada y devuelve una política de IAM como salida.

Utilice un autorizador de Lambda para implementar un esquema de autorización personalizado. El esquema puede utilizar los parámetros de la solicitud para determinar la identidad del intermediario o utilizar una estrategia de autenticación de token de portador como OAuth o SAML. Cree un autorizador de Lambda en la consola de la API de REST de API Gateway mediante la AWS CLI o un SDK de AWS.

Flujo del trabajo de autorización del autorizador de Lambda

El siguiente diagrama muestra el flujo de trabajo de autorización para un autorizador de Lambda.



Flujo de trabajo de autorización de Lambda en API Gateway

1. El cliente llama a un método en una API de API Gateway, pasando un token de portador o parámetros de solicitud.
2. API Gateway comprueba si la solicitud de método está configurada con un autorizador de Lambda. Si es así, API Gateway llama a la función de Lambda.
3. La función de Lambda autentica al intermediario. La función puede autenticarse de una de las siguientes formas:
   - Llamando a un proveedor de OAuth para obtener un token de acceso de OAuth.
   - Llamando a un proveedor de SAML para obtener una aserción de SAML.
   - Generando una política de IAM basada en los valores de los parámetros de la solicitud.
   - Recuperando credenciales de una base de datos.
4. La función de Lambda devuelve una política de IAM y un identificador de entidad principal. Si la función de Lambda no devuelve esa información, se produce un error en la llamada.
5. API Gateway evalúa la política de IAM.

- Si se deniega el acceso, API Gateway devuelve un código de estado HTTP apropiado, como por ejemplo `403 ACCESS_DENIED`.
- Si se permite el acceso, API Gateway invoca el método.

  Si habilita el almacenamiento en caché de la autorización, API Gateway almacena en caché la política de modo que no es necesario volver a invocar la función del autorizador de Lambda. Asegúrese de que la política se aplique a todos los recursos y métodos de la API.

Puede personalizar las respuestas de la puerta de enlace `403 ACCESS_DENIED` o `401 UNAUTHORIZED`. Para obtener más información, consulte the section called “Respuestas de gateway”.

Elección de un tipo de autorizador de Lambda

Hay dos tipos de autorizadores Lambda:

Autorizador de Lambda basado en parámetros de solicitud (autorizador **`REQUEST`**)

Un autorizador `REQUEST` recibe la identidad de la persona que llama en una combinación de encabezados, parámetros de cadenas de consulta, `stageVariables` y variables `$context`. Puede usar un autorizador `REQUEST` para crear políticas detalladas basadas en la información de varios orígenes de identidad, como las variables de contexto `$context.path` y `$context.httpMethod`.

Si activa el almacenamiento en caché de autorizaciones para un autorizador `REQUEST`, API Gateway verifica que todos los orígenes de identidad especificados estén presentes en la solicitud. Si falta algún origen de identidad especificado, o bien es nulo o está vacío, API Gateway devolverá una respuesta HTTP `401 Unauthorized` sin llamar a la función del autorizador de Lambda. Cuando se definen varios orígenes de identidad, se utilizan todos para obtener la clave de caché del autorizador, manteniendo el mismo orden. Puede definir una clave de caché detallada mediante el uso de varios orígenes de identidad.

Si cambia cualquiera de las partes de la clave de caché y vuelve a implementar su API, el autorizador descartará el documento de políticas almacenado en caché y generará otro nuevo.

Si desactiva el almacenamiento en caché de autorizaciones para un autorizador `REQUEST`, API Gateway pasa directamente la solicitud a la función de Lambda.

Autorizador de Lambda basado en token (autorizador **`TOKEN`**)

Un autorizador `TOKEN` recibe la identidad del intermediario en un token de portador, como, por ejemplo, un JSON Web Token (JWT) o un token de OAuth.

Si activa el almacenamiento en caché de autorizaciones para un autorizador `TOKEN`, el nombre del encabezado especificado en el origen del token pasará a ser la clave de caché.

Además, puede utilizar la validación del token para introducir una instrucción RegEx. API Gateway realiza la validación inicial del token de entrada con respecto a esta expresión e invoca la función del autorizador de Lambda tras una validación correcta. Esto ayuda a reducir las llamadas a la API.

La propiedad `IdentityValidationExpression` solo es compatible con autorizadores `TOKEN`. Para obtener más información, consulte the section called "`x-amazon-apigateway-authorizer`".

> **Note**
>
> Le recomendamos que utilice un autorizador `REQUEST` para controlar el acceso a su API. Puede controlar el acceso a su API en función de varios orígenes de identidad al utilizar un autorizador `REQUEST`, en comparación con un único origen de identidad al utilizar un autorizador `TOKEN`. Además, puede separar las claves de caché utilizando varios orígenes de identidad para un autorizador `REQUEST`.

Ejemplo de función de Lambda con un autorizador **`REQUEST`**

El siguiente código de ejemplo crea una función del autorizador de Lambda que permite realizar una solicitud si el encabezado `HeaderAuth1`, el parámetro de consulta `QueryString1` y la variable de etapa de `StageVar1` proporcionados por el cliente coinciden con los valores especificados de `headerValue1`, `queryValue1` y `stageValue1` respectivamente.

Node.js

```
// A simple request-based authorizer example to demonstrate how to use request
// parameters to allow or deny a request. In this example, a request is
// authorized if the client-supplied HeaderAuth1 header, QueryString1
// query parameter, and stage variable of StageVar1 all match
// specified values of 'headerValue1', 'queryValue1', and 'stageValue1',
```

```
// respectively.

export const handler = function(event, context, callback) {
    console.log('Received event:', JSON.stringify(event, null, 2));

    // Retrieve request parameters from the Lambda function input:
    var headers = event.headers;
    var queryStringParameters = event.queryStringParameters;
    var pathParameters = event.pathParameters;
    var stageVariables = event.stageVariables;

    // Parse the input for the parameter values
    var tmp = event.methodArn.split(':');
    var apiGatewayArnTmp = tmp[5].split('/');
    var awsAccountId = tmp[4];
    var region = tmp[3];
    var restApiId = apiGatewayArnTmp[0];
    var stage = apiGatewayArnTmp[1];
    var method = apiGatewayArnTmp[2];
    var resource = '/'; // root resource
    if (apiGatewayArnTmp[3]) {
        resource += apiGatewayArnTmp[3];
    }

    // Perform authorization to return the Allow policy for correct parameters and
    // the 'Unauthorized' error, otherwise.


    if (headers.HeaderAuth1 === "headerValue1"
        && queryStringParameters.QueryString1 === "queryValue1"
        && stageVariables.StageVar1 === "stageValue1") {
        callback(null, generateAllow('me', event.methodArn));
    }  else {
        callback(null, generateDeny('me', event.methodArn));
    }
}

// Help function to generate an IAM policy
var generatePolicy = function(principalId, effect, resource) {
    // Required output:
    var authResponse = {};
    authResponse.principalId = principalId;
    if (effect && resource) {
        var policyDocument = {};
```

```
        policyDocument.Version = '2012-10-17'; // default version
        policyDocument.Statement = [];
        var statementOne = {};
        statementOne.Action = 'execute-api:Invoke'; // default action
        statementOne.Effect = effect;
        statementOne.Resource = resource;
        policyDocument.Statement[0] = statementOne;
        authResponse.policyDocument = policyDocument;
    }
    // Optional output with custom properties of the String, Number or Boolean type.
    authResponse.context = {
        "stringKey": "stringval",
        "numberKey": 123,
        "booleanKey": true
    };
    return authResponse;
}

var generateAllow = function(principalId, resource) {
    return generatePolicy(principalId, 'Allow', resource);
}

var generateDeny = function(principalId, resource) {
    return generatePolicy(principalId, 'Deny', resource);
}
```

Python

```
# A simple request-based authorizer example to demonstrate how to use request
# parameters to allow or deny a request. In this example, a request is
# authorized if the client-supplied headerauth1 header, QueryString1
# query parameter, and stage variable of StageVar1 all match
# specified values of 'headerValue1', 'queryValue1', and 'stageValue1',
# respectively.

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

    # Retrieve request parameters from the Lambda function input:
    headers = event['headers']
    queryStringParameters = event['queryStringParameters']
    pathParameters = event['pathParameters']
    stageVariables = event['stageVariables']
```

```
    # Parse the input for the parameter values
    tmp = event['methodArn'].split(':')
    apiGatewayArnTmp = tmp[5].split('/')
    awsAccountId = tmp[4]
    region = tmp[3]
    restApiId = apiGatewayArnTmp[0]
    stage = apiGatewayArnTmp[1]
    method = apiGatewayArnTmp[2]
    resource = '/'

    if (apiGatewayArnTmp[3]):
        resource += apiGatewayArnTmp[3]

    # Perform authorization to return the Allow policy for correct parameters
    # and the 'Unauthorized' error, otherwise.

    if (headers['HeaderAuth1'] == "headerValue1" and
 queryStringParameters['QueryString1'] == "queryValue1" and
 stageVariables['StageVar1'] == "stageValue1"):
        response = generateAllow('me', event['methodArn'])
        print('authorized')
        return response
    else:
        print('unauthorized')
        response = generateDeny('me', event['methodArn'])
        return response
    # Help function to generate IAM policy


def generatePolicy(principalId, effect, resource):
    authResponse = {}
    authResponse['principalId'] = principalId
    if (effect and resource):
        policyDocument = {}
        policyDocument['Version'] = '2012-10-17'
        policyDocument['Statement'] = []
        statementOne = {}
        statementOne['Action'] = 'execute-api:Invoke'
        statementOne['Effect'] = effect
        statementOne['Resource'] = resource
        policyDocument['Statement'] = [statementOne]
        authResponse['policyDocument'] = policyDocument
```

```
    authResponse['context'] = {
        "stringKey": "stringval",
        "numberKey": 123,
        "booleanKey": True
    }

    return authResponse


def generateAllow(principalId, resource):
    return generatePolicy(principalId, 'Allow', resource)


def generateDeny(principalId, resource):
    return generatePolicy(principalId, 'Deny', resource)
```

En este ejemplo, la función del autorizador de Lambda comprueba los parámetros de entrada y actúa como se indica a continuación:

- Si todos los valores de los parámetros necesarios coinciden con los valores previstos, la función del autorizador devuelve una respuesta HTTP `200 OK` y una política de IAM que tiene un aspecto similar al siguiente, y la solicitud del método se lleva a cabo correctamente:

  JSON

  ```
  {
    "Version":"2012-10-17",
    "Statement": [
      {
        "Action": "execute-api:Invoke",
        "Effect": "Allow",
        "Resource": "arn:aws:execute-api:us-east-1:123456789012:ivdtdhp7b5/
  ESTestInvoke-stage/GET/"
      }
    ]
  }
  ```

- De lo contrario, la función del autorizador devuelve una respuesta HTTP `401 Unauthorized` y la solicitud al método produce un error.

Además de devolver una política de IAM, la función del autorizador de Lambda también debe devolver el identificador principal del intermediario. Opcionalmente, puede devolver un objeto `context`, que contiene información adicional que puede pasarse al backend de integración. Para obtener más información, consulte Salida de un autorizador de Lambda de API Gateway.

En el código de producción, es posible que necesite autenticar al usuario antes de conceder la autorización. En tal caso, puede agregar la lógica de autenticación a la función de Lambda llamando a un proveedor de autenticación de la forma que se indica en la documentación de ese proveedor.

Ejemplo de función de Lambda con un autorizador **`TOKEN`**

El siguiente código de ejemplo crea una función del autorizador de Lambda `TOKEN` que permite al intermediario invocar un método si el valor del token proporcionado por el cliente es `allow`. El intermediario no puede invocar la solicitud si el valor del token es `deny`. Si el valor del token es `unauthorized` o una cadena vacía, la función del autorizador devolverá una respuesta `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
```

En este ejemplo, cuando la API recibe una solicitud de método, API Gateway pasa el token de origen a esta función de autorizador de Lambda en el atributo `event.authorizationToken`. La función del autorizador de Lambda lee el token y actúa como se indica a continuación:

- Si el valor del token es `allow`, la función del autorizador devuelve una respuesta HTTP `200 OK` y una política de IAM que tiene un aspecto similar al siguiente, y la solicitud del método se lleva a cabo correctamente:

  JSON

  ```
  {
    "Version":"2012-10-17",
    "Statement": [
      {
        "Action": "execute-api:Invoke",
        "Effect": "Allow",
        "Resource": "arn:aws:execute-api:us-east-1:123456789012:ivdtdhp7b5/
  ESTestInvoke-stage/GET/"
      }
    ]
  }
  ```

- Si el valor del token es `deny`, la función del autorizador devolverá una respuesta HTTP `200 OK` y una política `Deny` de IAM similar a la siguiente y se producirá un error en la solicitud del método:

  JSON

  ```
  {
    "Version":"2012-10-17",
    "Statement": [
      {
        "Action": "execute-api:Invoke",
        "Effect": "Deny",
        "Resource": "arn:aws:execute-api:us-east-1:123456789012:ivdtdhp7b5/
  ESTestInvoke-stage/GET/"
      }
    ]
  ```

```
}
```

> **Note**
>
> Fuera del entorno de prueba, API Gateway devuelve una respuesta HTTP `403 Forbidden` y la solicitud del método produce un error.

- Si el valor del token es `unauthorized` o una cadena vacía, la función del autorizador devolverá una respuesta HTTP `401 Unauthorized` y la llamada al método producirá un error.
- Si el token es cualquier otra cosa, el cliente recibirá una respuesta `500 Invalid token` y la llamada al método producirá un error.

Además de devolver una política de IAM, la función del autorizador de Lambda también debe devolver el identificador principal del intermediario. Opcionalmente, puede devolver un objeto `context`, que contiene información adicional que puede pasarse al backend de integración. Para obtener más información, consulte Salida de un autorizador de Lambda de API Gateway.

En el código de producción, es posible que necesite autenticar al usuario antes de conceder la autorización. En tal caso, puede agregar la lógica de autenticación a la función de Lambda llamando a un proveedor de autenticación de la forma que se indica en la documentación de ese proveedor.

Ejemplos adicionales de funciones del autorizador de Lambda

En la siguiente lista, se muestran ejemplos adicionales de funciones del autorizador de Lambda. Puede crear una función de Lambda en la misma cuenta desde la que creó la API o en una cuenta diferente.

Para las funciones de Lambda del ejemplo anterior, puede utilizar el AWSLambdaBasicExecutionRole integrado, ya que estas funciones no llaman a otros servicios de AWS. Si su función de Lambda llama a otros servicios de AWS, tendrá que asignar un rol de ejecución de IAM para la función de Lambda. Para crear el rol, siga las instrucciones de Rol de ejecución de AWS Lambda.

Ejemplos adicionales de funciones del autorizador de Lambda

- Para ver una aplicación de ejemplo, consulte Open Banking Brazil - Ejemplos de autorización en GitHub.
- Para ver más ejemplos de funciones de Lambda, consulte aws-apigateway-lambda-authorizer-blueprints en GitHub.

- Puede crear un autorizador de Lambda que autentique a los usuarios mediante grupos de usuarios de Amazon Cognito y autorice a los intermediarios en función de un almacén de políticas mediante permisos verificados. Para obtener más información, consulte the section called “Control del acceso en función de los atributos de una identidad con Verified Permissions”.
- La consola de Lambda ofrece un proyecto de Python, que se puede utilizar eligiendo Use a blueprint (Utilizar un proyecto) y eligiendo el proyecto api-gateway-authorizer-python.

Configuración de un autorizador de Lambda de API Gateway

Tras crear una función de Lambda, debe configurarla como un autorizador para su API. A continuación, configure su método para invocar su autorizador de Lambda y determinar si un intermediario puede invocar su método. Puede crear una función de Lambda en la misma cuenta desde la que creó la API o en una cuenta diferente.

Puede probar su autorizador de Lambda mediante las herramientas integradas en la consola de API Gateway o con Postman. Para obtener instrucciones sobre cómo usar Postman para probar la función del autorizador de Lambda, consulte the section called “Llamada a una API con autorizadores de Lambda”.

Configuración de un autorizador de Lambda (consola)

El siguiente procedimiento muestra cómo crear un autorizador de Lambda en la consola de la API de REST de API Gateway. Para obtener más información sobre los distintos tipos de autorizadores de Lambda, consulte the section called “Elección de un tipo de autorizador de Lambda”.

REQUEST authorizer

Configuración de un autorizador de Lambda **`REQUEST`**

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Seleccione una API y, a continuación, elija Autorizadores.
3. Elija Crear autorizador.
4. En Nombre del autorizador, ingrese un nombre para el autorizador.
5. En Tipo de autorizador, seleccione Lambda.
6. En Función de Lambda, seleccione la Región de AWS en la que creó la función de autorizador de Lambda y, a continuación, ingrese el nombre de la función.
7. Deje en blanco Rol de invocación de Lambda para permitir que la consola de la API de REST de API Gateway defina una política basada en recursos. La política concede permisos a

API Gateway para invocar la función del autorizador de Lambda. También puede escribir el nombre de un rol de IAM para permitir que API Gateway invoque la función del autorizador de Lambda. Para ver un rol de ejemplo, consulte Crear un rol de IAM asumible.

8. En Carga de evento de Lambda, seleccione Solicitud.
9. En Tipo de origen de identidades, seleccione un tipo de parámetro. Los tipos de parámetro admitidos son `Header`, `Query string`, `Stage variable` y `Context`. Para agregar más orígenes de identidades, elija Agregar parámetro.
10. Para almacenar en caché la política de autorización generada por el autorizador, mantenga activado Almacenamiento en caché de la autorización. Cuando se activa el almacenamiento en caché de políticas, puede modificar el valor TTL. Si establece TTL en cero se desactivará el almacenamiento en caché de políticas.

    Si habilita el almacenamiento en caché, su autorizador debe devolver una política que se aplique a todos los métodos a través de una API. Para aplicar una política específica de un método, utilice las variables de contexto `$context.path` y `$context.httpMethod`.
11. Elija Crear autorizador.

TOKEN authorizer

Configuración de un autorizador de Lambda **`TOKEN`**

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Seleccione una API y, a continuación, elija Autorizadores.
3. Elija Crear autorizador.
4. En Nombre del autorizador, ingrese un nombre para el autorizador.
5. En Tipo de autorizador, seleccione Lambda.
6. En Función de Lambda, seleccione la Región de AWS en la que creó la función de autorizador de Lambda y, a continuación, ingrese el nombre de la función.
7. Deje en blanco Rol de invocación de Lambda para permitir que la consola de la API de REST de API Gateway defina una política basada en recursos. La política concede permisos a API Gateway para invocar la función del autorizador de Lambda. También puede escribir el nombre de un rol de IAM para permitir que API Gateway invoque la función del autorizador de Lambda. Para ver un rol de ejemplo, consulte Crear un rol de IAM asumible.
8. En Carga de evento de Lambda, seleccione Token.

9. En Origen del token, ingrese el nombre del encabezado que contiene el token de autorización. El intermediario debe incluir un encabezado con este nombre para enviar el token de autorización al autorizador de Lambda.
10. (Opcional) Si lo desea, en Validación del token, introduzca una instrucción RegEx. API Gateway realiza la validación inicial del token de entrada contra esta expresión e invoca el autorizador tras una validación correcta.
11. Para almacenar en caché la política de autorización generada por el autorizador, mantenga activado Almacenamiento en caché de la autorización. Cuando el almacenamiento en caché de las políticas esté habilitado, el nombre del encabezado especificado en Origen del token pasará a ser la clave de caché. Cuando se activa el almacenamiento en caché de políticas, puede modificar el valor TTL. Si establece TTL en cero se desactivará el almacenamiento en caché de políticas.

    Si habilita el almacenamiento en caché, su autorizador debe devolver una política que se aplique a todos los métodos a través de una API. Para aplicar una política específica de un método, puede desactivar Almacenamiento en caché de autorización.
12. Elija Crear autorizador.

Después de crear el autorizador de Lambda, puede probarlo. El siguiente procedimiento muestra cómo probar el autorizador de Lambda.

REQUEST authorizer

Prueba de un autorizador de Lambda **`REQUEST`**

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Seleccione el nombre del autorizador.
3. En Probar autorizador, introduzca un valor para su origen de identidad.

    Si utiliza la the section called “Ejemplo de función de Lambda con un autorizador REQUEST”, haga lo siguiente:

    a. Seleccione Encabezado e ingrese **`headerValue1`** y, a continuación, elija Agregar parámetro.

    b. En Tipo de origen de identidades, seleccione Cadena de consulta e ingrese **`queryValue1`** y, a continuación, elija Agregar parámetro.

c. En Tipo de origen de identidades, seleccione Variable de etapa e ingrese **`stageValue1`**.

No puede modificar las variables de contexto para la invocación de la prueba, pero sí puede modificar la plantilla de eventos de prueba del Autorizador de API Gateway para la función de Lambda. A continuación, puede probar la función del autorizador de Lambda con variables de contexto modificadas. Para obtener más información, consulte Prueba de funciones de Lambda en la consola en la Guía para desarrolladores de AWS Lambda.

4. Elija Probar el autorizador.

TOKEN authorizer

Prueba de un autorizador de Lambda **`TOKEN`**

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Seleccione el nombre del autorizador.
3. En Probar autorizador, introduzca un valor para su token.

   Si utiliza la the section called “Ejemplo de función de Lambda con un autorizador TOKEN”, haga lo siguiente:

   - En authorizationToken, especifique **`allow`**.
4. Elija Probar el autorizador.

   Si el autorizador de Lambda rechaza correctamente una solicitud en el entorno de prueba, la prueba produce una respuesta HTTP `200 OK`. Sin embargo, fuera del entorno de prueba, API Gateway devuelve una respuesta HTTP `403 Forbidden` y la solicitud del método produce un error.

Configuración de un autorizador de Lambda (AWS CLI)

El siguiente comando create-authorizer muestra cómo crear un autorizador de Lambda mediante la AWS CLI.

REQUEST authorizer

El siguiente comando create-authorizer permite crear un autorizador de `REQUEST`, y utiliza el encabezado `Authorizer` y la variable de contexto `accountId` como orígenes de identidad:

```
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

El siguiente comando create-authorizer permite crear un autorizados de `TOKEN`, y utiliza el encabezado `Authorization` como origen de identidad:

```
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
```

Después de crear el autorizador de Lambda, puede probarlo. El siguiente comando test-invoke-authorizer permite probar un autorizador de Lambda:

```
aws apigateway test-invoke-authorizer --rest-api-id 1234123412 \
   --authorizer-id efg1234 \
   --headers Authorization='Value'
```

Configuración de un método para utilizar un autorizador de Lambda (consola)

Después de configurar el autorizador de Lambda, debe asociarlo a un método para la API. Si el autorizador utiliza el almacenamiento en caché de la autorización, asegúrese de actualizar la política para controlar el acceso del método adicional.

Para configurar un método de API para que use un autorizador de Lambda

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Seleccione una API.
3. Elija Recursos y seleccione un método nuevo o elija un método existente.
4. En la pestaña Solicitud de método, en Configuración de solicitud de método, elija Editar.
5. En Autorizador, en el menú desplegable, seleccione el autorizador de Lambda que acaba de crear.
6. (Opcional) Si quiere pasar el token de autorización al backend, elija los encabezados de solicitud HTTP. Elija Agregar encabezado y, a continuación, agregue el nombre del encabezado de autorización. En Nombre, escriba el nombre de encabezado que coincida con el nombre de Origen del token que especificó al crear el autorizador de Lambda para la API. Este paso no se aplica a los autorizadores `REQUEST`.
7. Seleccione Save.
8. Elija Deploy API (Implementar API) para implementar la API en una etapa. Para un autorizador `REQUEST` que utiliza variables de etapa, también debe definir las variables de etapa necesarias y especificar los valores en la página Etapas.

Configuración de un método para utilizar un autorizador de Lambda (AWS CLI)

Después de configurar el autorizador de Lambda, debe asociarlo a un método para la API. Puede crear un método nuevo o utilizar una operación de parche para asociar un autorizador a un método existente. Si el autorizador utiliza el almacenamiento en caché de la autorización, asegúrese de actualizar la política para controlar el acceso del método adicional.

El siguiente comando put-method permite crear un método nuevo que utilice un autorizador de Lambda:

```
aws apigateway put-method --rest-api-id 1234123412 \
  --resource-id a1b2c3 \
  --http-method PUT \
  --authorization-type CUSTOM \
  --authorizer-id efg1234
```

El siguiente comando update-method permite actualizar un método existente para que utilice un autorizador de Lambda:

```
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"
```

Entrada de un autorizador de Lambda de API Gateway

En la siguiente sección se explica el formato de la entrada de API Gateway para un autorizador de Lambda.

Formato de entrada de **`TOKEN`**

Para un autorizador de Lambda (que anteriormente se denominaba autorizador personalizado) del tipo `TOKEN`, debe especificar un encabezado personalizado como Token Source (Origen del token) al configurar el autorizador para una API. El cliente de la API debe transmitir el token de autorización necesario en ese encabezamiento en la solicitud de entrada. Tras recibir la solicitud del método de entrada, API Gateway extrae el token del encabezado personalizado. A continuación, transmite el token como propiedad `authorizationToken` del objeto `event` de la función de Lambda, además del ARN del método como propiedad `methodArn`:

```
{
    "type":"TOKEN",
    "authorizationToken":"{caller-supplied-token}",
    "methodArn":"arn:aws:execute-api:{regionId}:{accountId}:{apiId}/{stage}/{httpVerb}/
[{resource}/[{child-resources}]]"
}
```

En este ejemplo, la propiedad `type` especifica el tipo de autorizador, que es un autorizador `TOKEN`. *`{caller-supplied-token}`* procede del encabezado de autorización de una solicitud del cliente y puede ser cualquier valor de cadena. `methodArn` es el ARN de la solicitud del método de entrada y lo rellena API Gateway de acuerdo con la configuración del autorizador de Lambda.

Formato de entrada de **`REQUEST`**

Para un autorizador de Lambda del tipo `REQUEST`, API Gateway transmite los parámetros de solicitud a la función de Lambda del autorizador como parte del objeto `event`. Los parámetros de solicitud incluyen encabezados, parámetros de ruta, parámetros de cadenas de consulta, variables de etapa y algunas variables de contexto de solicitud. El intermediario de la API puede definir parámetros de

ruta, encabezados y parámetros de cadenas de consulta. El desarrollador de la API debe establecer las variables de etapa durante la implementación de la API y API Gateway proporciona el contexto de la solicitud en tiempo de ejecución.

> **Note**
>
> Los parámetros de la ruta se pueden pasar como parámetros de solicitud a la función del autorizador de Lambda pero no se pueden usar como fuentes de identidad.

El siguiente ejemplo muestra una entrada de un autorizador `REQUEST` para un método de la API (`GET /request`) con una integración de proxy:

```
{
  "type": "REQUEST",
  "methodArn": "arn:aws:execute-api:us-east-1:123456789012:abcdef123/test/GET/request",
  "resource": "/request",
  "path": "/request",
  "httpMethod": "GET",
  "headers": {
    "X-AMZ-Date": "20170718T062915Z",
    "Accept": "*/*",
    "HeaderAuth1": "headerValue1",
    "CloudFront-Viewer-Country": "US",
    "CloudFront-Forwarded-Proto": "https",
    "CloudFront-Is-Tablet-Viewer": "false",
    "CloudFront-Is-Mobile-Viewer": "false",
    "User-Agent": "..."
  },
  "queryStringParameters": {
    "QueryString1": "queryValue1"
  },
  "pathParameters": {},
  "stageVariables": {
    "StageVar1": "stageValue1"
  },
  "requestContext": {
    "path": "/request",
    "accountId": "123456789012",
    "resourceId": "05c7jb",
    "stage": "test",
    "requestId": "...",
```

```
    "identity": {
      "apiKey": "...",
      "sourceIp": "...",
      "clientCert": {
        "clientCertPem": "CERT_CONTENT",
        "subjectDN": "www.example.com",
        "issuerDN": "Example issuer",
        "serialNumber": "a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1",
        "validity": {
          "notBefore": "May 28 12:30:02 2019 GMT",
          "notAfter": "Aug  5 09:36:04 2021 GMT"
        }
      }
    },
    "resourcePath": "/request",
    "httpMethod": "GET",
    "apiId": "abcdef123"
  }
}
```

`requestContext` es un mapa de pares de clave-valor y se corresponde con la variable $context. Su resultado depende de la API.

API Gateway podría agregar nuevas claves al mapa. Para obtener más información sobre la entrada de la función de Lambda en una integración de proxy, consulte Formato de entrada de una función de Lambda para la integración de proxy.

Salida de un autorizador de Lambda de API Gateway

La salida de una función de autorizador de Lambda es un objeto similar a un diccionario que debe incluir el identificador de la entidad principal (`principalId`) y un documento de política (`policyDocument`) que contenga una lista de instrucciones de política. La salida también puede incluir un mapa `context` que contenga pares de clave-valor. Si la API utiliza un plan de uso (`apiKeySource` se ha establecido en `AUTHORIZER`), la función de autorizador de Lambda debe devolver una de las claves de API del plan de uso como valor de la propiedad `usageIdentifierKey`.

A continuación se muestra un ejemplo de esta salida.

JSON

```
{
  "principalId": "yyyyyyyy",
  "policyDocument": {
    "Version":"2012-10-17",
    "Statement": [
      {
        "Action": "execute-api:Invoke",
        "Effect": "Allow|Deny",
        "Resource": "arn:aws:execute-
api:{regionId}:{accountId}:{apiId}/{stage}/{httpVerb}/[{resource}/[{child-
resources}]]"
      }
    ]
  },
  "context": {
    "stringKey": "value",
    "numberKey": "1",
    "booleanKey": "true"
  },
  "usageIdentifierKey": "{api-key}"
}
```

Aquí, una instrucción de política especifica si se permite o deniega (`Effect`) al servicio de ejecución de API Gateway invocar (`Action`) el método de API especificado (`Resource`). Es posible que tenga que controlar el acceso a varios recursos en función del autorizador. Puede utilizar un carácter comodín (`*`) para especificar un tipo de recurso (método). Para obtener información sobre la configuración de políticas válidas para llamar a una API, consulte Referencia de instrucciones de políticas de IAM para ejecutar la API en API Gateway.

En el caso de los ARN del método con autorización habilitada (por ejemplo, `arn:aws:execute-api:{regionId}:{accountId}:{apiId}/{stage}/{httpVerb}/[{resource}/[{child-resources}]]`), la longitud máxima es de 1600 bytes. Los valores de parámetros de ruta, cuyo tamaño se determina en tiempo de ejecución, pueden provocar que la longitud del ARN supere el límite. Cuando esto ocurra, el cliente de la API recibirá una respuesta `414 Request URI too long`.

Además, el ARN de recurso, tal y como se muestra en la instrucción de política de salida que ha realizado el autorizador, está limitado actualmente a 512 caracteres. Por este motivo, no debe utilizar la URI con un token JWT que tenga una longitud considerable en una URI de la solicitud. Puede transferir de manera segura el token JWT en un encabezado de solicitud.

Puede tener acceso al valor `principalId` de una plantilla de asignación utilizando la variable `$context.authorizer.principalId`. Esto resulta útil si desea transmitir el valor al backend. Para obtener más información, consulte Variables de contexto para transformaciones de datos.

Puede obtener acceso al valor de `stringKey`, `numberKey` o `booleanKey` (por ejemplo, `"value"`, `"1"` o `"true"`) del mapa `context` en una plantilla de mapeo llamando a `$context.authorizer.stringKey`, `$context.authorizer.numberKey` o `$context.authorizer.booleanKey`, respectivamente. Los valores devueltos se representan en forma de cadena. Observe que no puede establecer un objeto o matriz JSON como un valor válido de ninguna clave en el mapa `context`.

Puede utilizar el mapa `context` para devolver las credenciales almacenadas en caché del autorizador al backend utilizando una plantilla de mapeo de una solicitud de integración. Esto permite que el backend proporcione una mejor experiencia de usuario mediante la utilización de las credenciales almacenadas en caché para reducir la necesidad de obtener acceso a las claves secretas y abrir los tokens de autorización para cada solicitud.

En el caso de la integración de proxy de Lambda, API Gateway transmite el objeto `context` de un autorizador de Lambda directamente a la función de Lambda del backend como parte de la entrada `event`. Puede recuperar los pares de clave-valor de `context` en la función de Lambda llamando a `$event.requestContext.authorizer.key`.

`{api-key}` representa una clave de API en el plan de uso de la etapa de API. Para obtener más información, consulte the section called “Planes de uso”.

El siguiente ejemplo muestra la salida del ejemplo de autorizador de Lambda. El resultado de ejemplo contiene una instrucción de política para bloquear (`Deny`) las llamadas al método `GET` para la etapa `dev` de una API (`ymy8tbxw7b`) de una cuenta de AWS (`123456789012`).

JSON

```
{
  "principalId": "user",
```

```
  "policyDocument": {
    "Version":"2012-10-17",
    "Statement": [
      {
        "Action": "execute-api:Invoke",
        "Effect": "Deny",
        "Resource": "arn:aws:execute-api:us-west-2:123456789012:ymy8tbxw7b/dev/
GET/"
      }
    ]
  }
}
```

Llamada a una API con un autorizador de Lambda de API Gateway

Después de configurar el autorizador de Lambda (que anteriormente se denominaba autorizador personalizado); e implementar la API, debe probar la API con el autorizador de Lambda habilitado. Para ello, necesita un cliente REST, como cURL o Postman. En los siguientes ejemplos, usamos Postman.

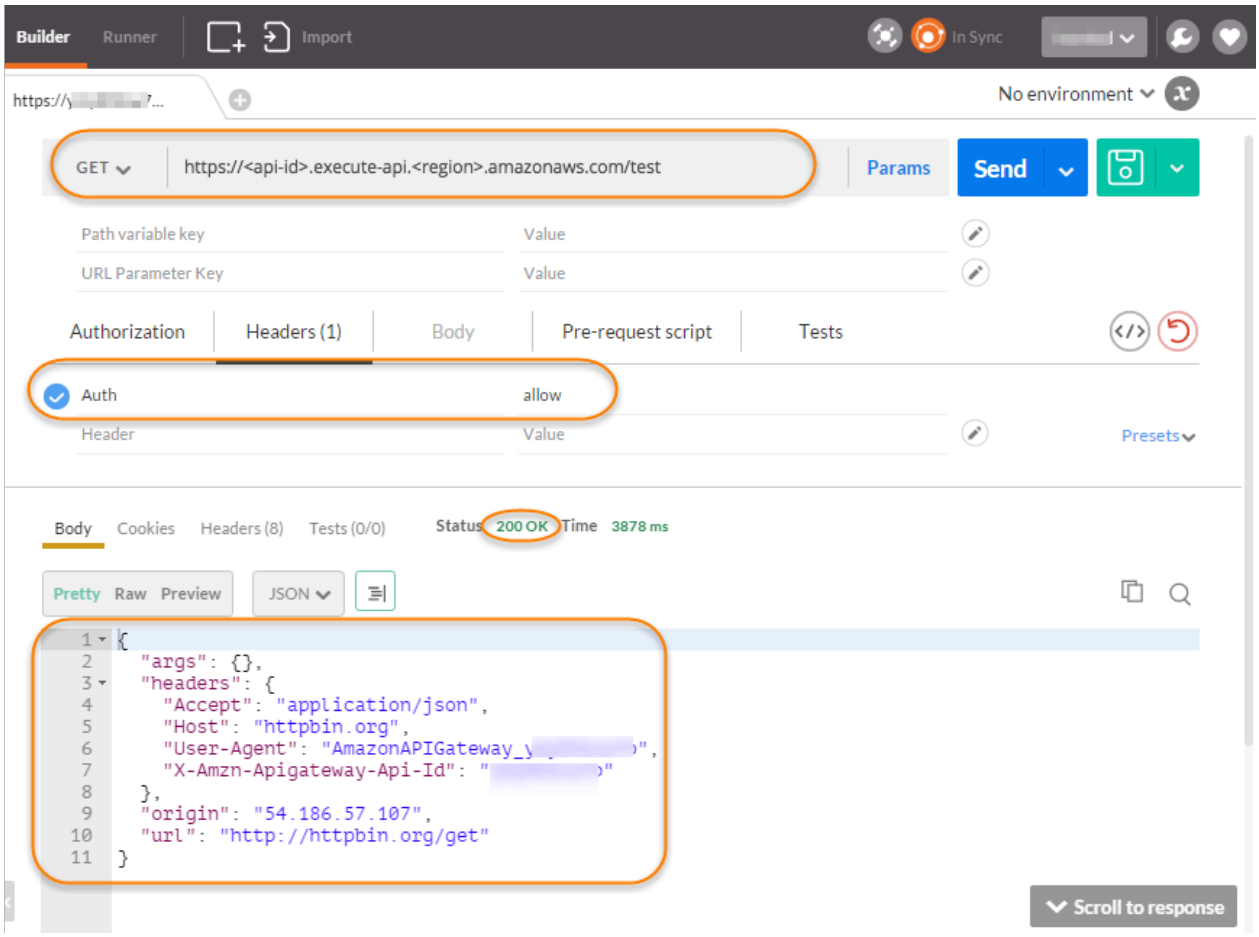
> **Note**
>
> Al llamar a un método habilitado para un autorizador, API Gateway no registra la llamada a CloudWatch si el token necesario para el autorizador `TOKEN` no se establece, es nulo o no es válido para la expresión Token validation expression (Expresión de validación de token) especificada. Del mismo modo, API Gateway no registra la llamada a CloudWatch si alguno de los orígenes de identidad necesarios para el autorizador `REQUEST` no está establecido, es nulo o está vacío.

A continuación, mostramos cómo utilizar Postman para llamar a la API o probarla con un autorizador de Lambda `TOKEN`. El método puede usarse para llamar a una API con un autorizador de Lambda `REQUEST` si se especifican los parámetros de ruta, encabezado o cadena de consulta necesarios de forma explícita.

Llamar a una API con el autorizador personalizado **`TOKEN`**

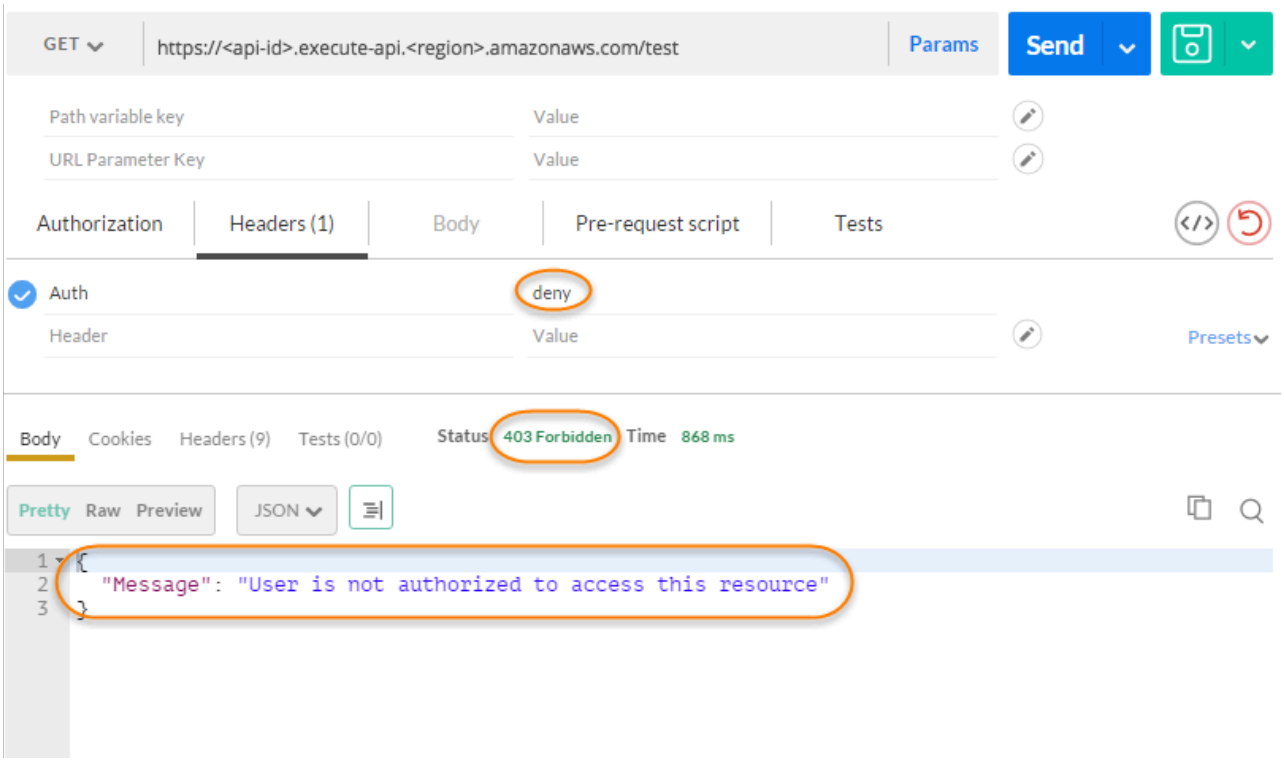
1. Abra Postman, elija el método GET y pegue el valor de Invoke URL (Invocar URL) de la API en el campo URL contiguo.

Agregue el encabezado de token de autorización de Lambda y establezca el valor en `allow`. Seleccione Send (enviar).



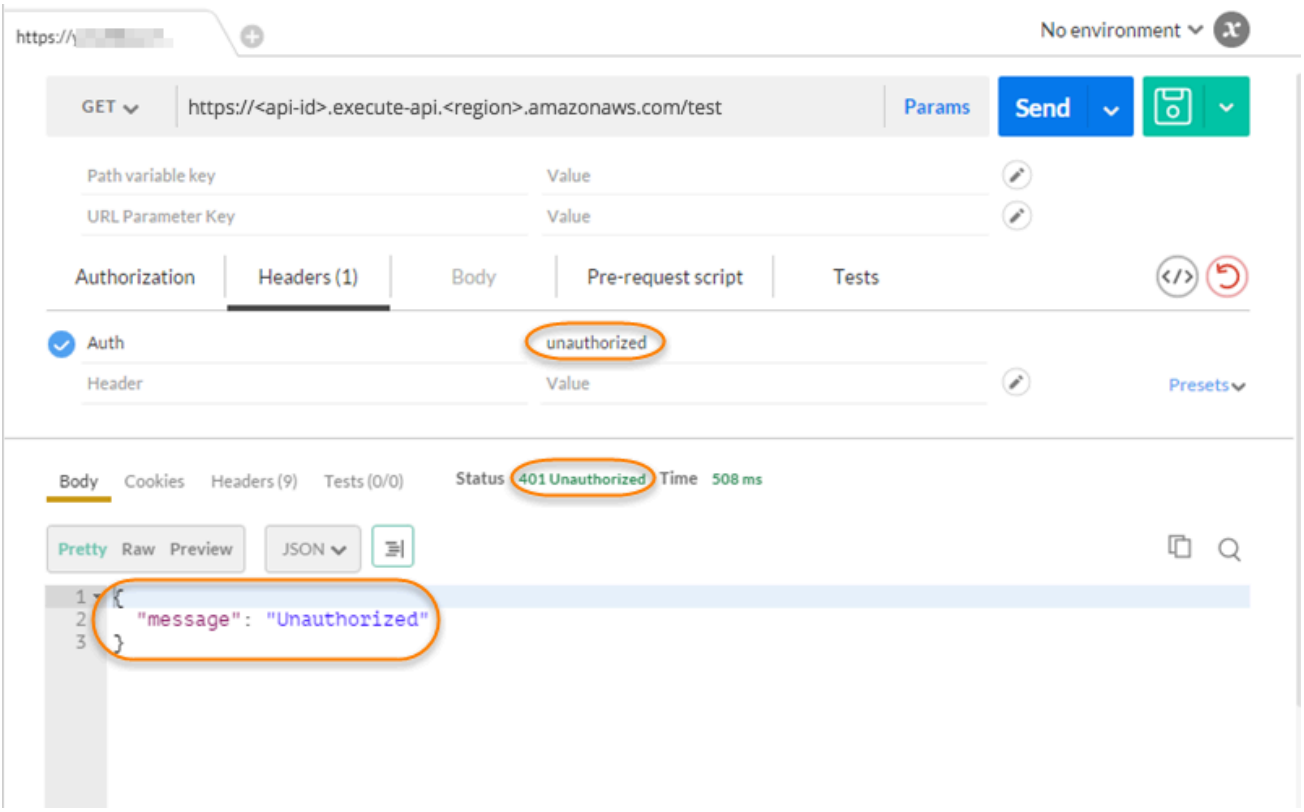
La respuesta muestra que el autorizador de Lambda de API Gateway devuelve una respuesta 200 OK y autoriza correctamente a la llamada para que tenga acceso al punto de conexión HTTP (http://httpbin.org/get) integrado en el método.

2. Sin salir de Postman, cambie el valor del encabezado de token de autorización de Lambda a `deny`. Seleccione Send (enviar).



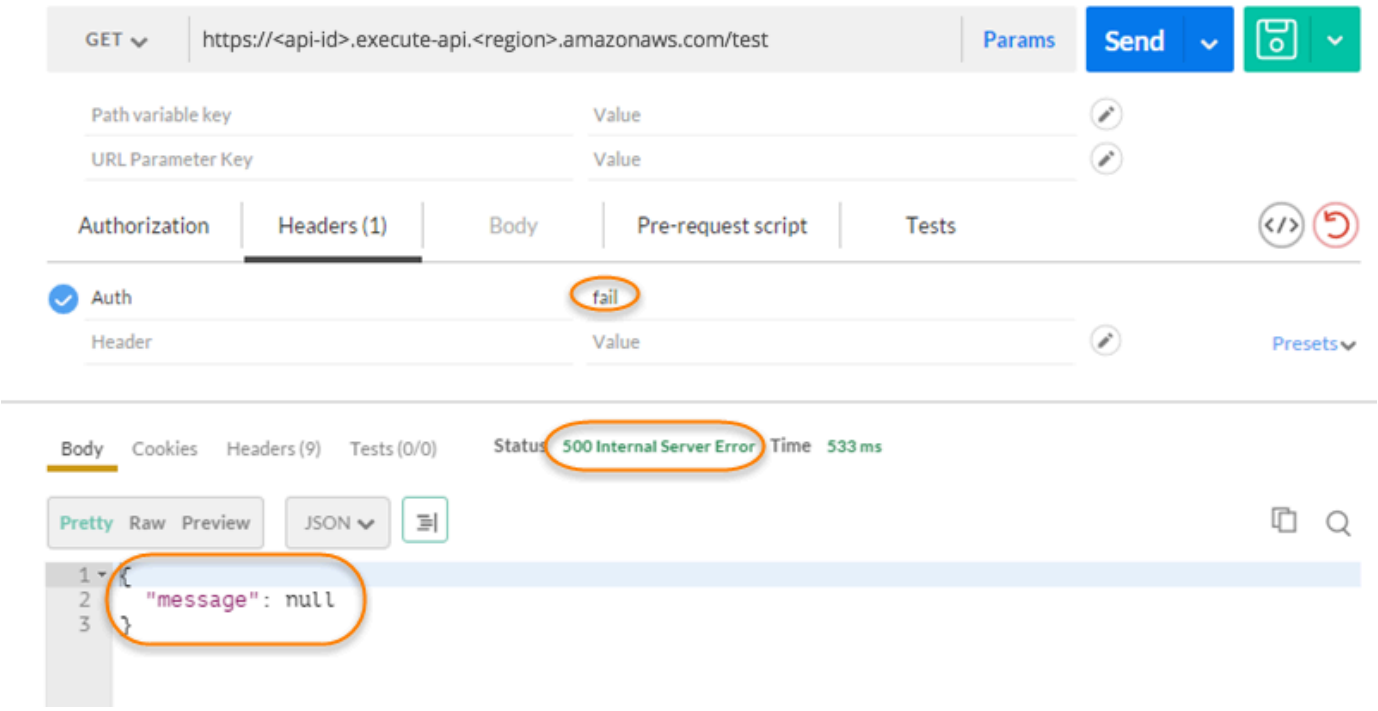
La respuesta muestra que el autorizador de Lambda de API Gateway devuelve una respuesta 403 Forbidden que no autoriza a la llamada para que tenga acceso al punto de conexión HTTP.

3. En Postman, cambie el valor del encabezado de token de autorización de Lambda a `unauthorized` y elija Send (Enviar).



La respuesta muestra que API Gateway devuelve una respuesta 401 Unauthorized sin autorizar la llamada para tener acceso al punto de conexión HTTP.

4. Ahora, cambie el valor del encabezado de token de autorización de Lambda a `fail`. Seleccione Send (enviar).



La respuesta muestra que API Gateway devuelve una respuesta 500 Internal Server Error sin autorizar la llamada para tener acceso al punto de conexión HTTP.

Configuración de un autorizador de Lambda de API Gateway entre cuentas

Ahora también se puede utilizar una función de AWS Lambda de otra cuenta de AWS como función de autorizador de API. Cada cuenta puede estar en cualquier región en la que Amazon API Gateway esté disponible. La función de autorizador de Lambda puede utilizar estrategias de autenticación de token al portador como OAuth o SAML. Esto permite administrar y compartir fácilmente de forma centralizada la función de autorizador de Lambda en distintas API de API Gateway.

En esta sección, mostramos cómo configurar una función de autorización de Lambda entre cuentas mediante la consola de Amazon API Gateway.

En estas instrucciones, se presupone que usted ya dispone de una API de API Gateway en una cuenta de AWS y de una función de autorizador de Lambda en otra cuenta.

Configurar un autorizador de Lambda entre cuentas mediante la consola de API Gateway

Inicie sesión en la consola de Amazon API Gateway en la cuenta que tiene la API y, a continuación, haga lo siguiente:

1. Elija la API y, a continuación, en el panel de navegación principal, elija Autorizadores.

2. Elija Crear autorizador.
3. En Nombre del autorizador, ingrese un nombre para el autorizador.
4. En Tipo de autorizador, seleccione Lambda.
5. En Función de Lambda, copie y pegue el ARN completo de la función del autorizador de Lambda que tiene en la segunda cuenta.

> **Note**
>
> En la consola de Lambda, puede encontrar el ARN de la función en la esquina superior derecha de la ventana de la consola.

6. Aparecerá una advertencia con una cadena de comandos `aws lambda add-permission`. La política concede permiso de API Gateway para invocar la función de Lambda del autorizador. Copie el comando y guárdelo para más adelante. Ejecute el comando después de crear el autorizador.
7. En Carga útil del evento de Lambda, seleccione Token para un autorizador `TOKEN` o Solicitud para un autorizador `REQUEST`.
8. Dependiendo de la elección del paso anterior, lleve a cabo alguna de las siguientes operaciones:

   a. Para la opción Token, haga lo siguiente:

   - En Origen del token, ingrese el nombre del encabezado que contiene el token de autorización. El cliente de la API debe incluir un encabezado con este nombre para enviar el token de autorización al autorizador de Lambda.
   - Si lo desea, en Validación del token, ingrese una instrucción RegEx. API Gateway realiza la validación inicial del token de entrada contra esta expresión e invoca el autorizador tras una validación correcta. Esto ayuda a reducir las llamadas a la API.
   - Para almacenar en caché la política de autorización generada por el autorizador, mantenga activado Almacenamiento en caché de la autorización. Cuando se activa el almacenamiento en caché de políticas, puede modificar el valor TTL. Si establece TTL en cero se desactivará el almacenamiento en caché de políticas. Cuando el almacenamiento en caché de las políticas esté habilitado, el nombre del encabezado especificado en Origen del token pasará a ser la clave de caché. Si se pasan varios valores a este encabezado de la solicitud, todos los valores se convertirán en la clave de caché y se conservará el orden.

> **Note**
>
> El valor de TTL predeterminado es de 300 segundos. El valor máximo es de 3600 segundos, este límite no se puede aumentar.

b. Para la opción Request (Solicitud), haga lo siguiente:

- En Tipo de origen de identidades, seleccione un tipo de parámetro. Los tipos de parámetro admitidos son `Header`, `Query string`, `Stage variable` y `Context`. Para agregar más orígenes de identidades, elija Agregar parámetro.
- Para almacenar en caché la política de autorización generada por el autorizador, mantenga activado Almacenamiento en caché de la autorización. Cuando se activa el almacenamiento en caché de políticas, puede modificar el valor TTL. Si establece TTL en cero se desactivará el almacenamiento en caché de políticas.

  API Gateway utiliza las fuentes de identidad especificadas como la clave de caché del autorizador de la solicitud. Cuando está habilitado el almacenamiento en caché, API Gateway llama a la función de Lambda del autorizador solo después de verificar correctamente que todas las fuentes de identidad especificadas estén presentes en tiempo de ejecución. Si falta un origen de identidad especificado, es nulo o está vacío, API Gateway devuelve una respuesta `401 Unauthorized` sin llamar a la función de Lambda del autorizador.

  Cuando se definen varias fuentes de identidad, se utilizan todas ellas para obtener la clave de caché del autorizador. Cambiar cualquiera de las partes de la clave de caché hace que el autorizador descarte el documento de políticas almacenado en caché y genere otro nuevo. Si se pasa un encabezado con varios valores en la solicitud, todos los valores se convertirán en la clave de caché y se conservará el orden.
- Cuando está desactivado el almacenamiento en caché, no es necesario especificar ningún origen de identidades.

> **Note**
>
> Para habilitar el almacenamiento en caché, su autorizador debe devolver una política que se aplique a todos los métodos a través de una API. Para aplicar una política específica de método, puede desactivar Almacenamiento en caché de la autorización.

9. Elija Crear autorizador.
10. Pegue la cadena de comandos `aws lambda add-permission` que copió durante un paso anterior en una ventana de la AWS CLI, que está configurada para la segunda cuenta. Reemplace `AUTHORIZER_ID` por el ID de su autorizador. Este concederá a la primera cuenta acceso a la función de autorizador de Lambda de la segunda cuenta.

Control del acceso en función de los atributos de una identidad con Verified Permissions

Utilice Amazon Verified Permissions para controlar el acceso a su API de API Gateway. Cuando utiliza API Gateway con Verified Permissions, Verified Permissions crea un autorizador de Lambda que utiliza decisiones de autorización detalladas para controlar el acceso a la API. Verified Permissions autoriza a los intermediarios según el esquema y las políticas del almacenamiento de políticas mediante el lenguaje de políticas de Cedar para definir permisos detallados para los usuarios de las aplicaciones. Para obtener más información, consulte Create a policy store with a connected API and identity provider en la Guía del usuario de Amazon Verified Permissions.

Verified Permissions admite grupos de usuarios de Amazon Cognito o proveedores de identidad OpenID Connect (OIDC) como orígenes de identidad. Verified Permissions presupone que la entidad principal se ha identificado y autenticado previamente. Verified Permissions solo es compatible con las API de REST regionales y optimizadas para la periferia.

Creación de un autorizador de Lambda mediante Verified Permissions

Verified Permissions crea un autorizador de Lambda para determinar si una entidad principal está autorizada a realizar una acción en la API. Cree la política de Cedar que Verified Permissions utiliza para realizar sus tareas de autorización.

En el siguiente ejemplo se muestra una política de Cedar que permite el acceso para invocar una API basada en el grupo de usuarios de Amazon Cognito, `us-east-1_ABC1234` para el grupo `developer` en el recurso `GET /users` de una API. Verified Permissions determina la pertenencia al grupo mediante el análisis del token portador para la identidad del intermediario.

```
permit(
  principal in MyAPI::UserGroup::"us-east-1_ABC1234|developer",
  action in [ MyAPI::Action::"get /users" ],
  resource
  );
```

De forma opcional, Verified Permissions puede asociar el autorizador a los métodos de la API. En las etapas de producción de la API, le recomendamos que no permita que Verified Permissions asocie el autorizador por usted.

En la siguiente lista se muestra cómo configurar Verified Permissions para asociar o no el autorizador de Lambda a la solicitud de los métodos de la API.

Asociación del autorizador por usted (Consola de administración de AWS)

Cuando elija Crear almacén de políticas en la consola de Verified Permissions, en la página Implementar integración de aplicaciones, elija Ahora.

Sin asociación del autorizador por usted (Consola de administración de AWS)

Cuando elija Crear almacén de políticas en la consola de Verified Permissions, en la página Implementar integración de aplicaciones, elija Más tarde.

Verified Permissions sigue creando un autorizador de Lambda por usted. El autorizador de Lambda comienza por `AVPAuthorizerLambda-`. Para obtener más instrucciones sobre cómo adjuntar el autorizador en un método, consulte the section called “Configuración de un método para utilizar un autorizador de Lambda (consola)”.

Asociación del autorizador por usted (CloudFormation)

En la plantilla CloudFormation generada por Verified Permissions, en la sección `Conditions`, establezca `"Ref": "shouldAttachAuthorizer"` a `true`.

Sin asociación del autorizador por usted (CloudFormation)

En la plantilla CloudFormation generada por Verified Permissions, en la sección `Conditions`, establezca `"Ref": "shouldAttachAuthorizer"` a `false`.

Verified Permissions sigue creando un autorizador de Lambda por usted. El autorizador de Lambda comienza por `AVPAuthorizerLambda-`. Para obtener más instrucciones sobre cómo adjuntar el autorizador en un método, consulte the section called “Configuración de un método para utilizar un autorizador de Lambda (AWS CLI)”.

Llamada a un autorizador de Lambda mediante Verified Permissions

Puede llamar a al autorizador de Lambda si proporciona una identidad o un token de acceso en el encabezado `Authorization`. Para obtener más información, consulte the section called “Llamada a una API con autorizadores de Lambda”.

API Gateway almacena en caché la política que devuelve el autorizador de Lambda durante 120 segundos. Puede modificar el TTL en la consola de API Gateway o mediante la AWS CLI.

Control del acceso a las API de REST con grupos de usuarios de Amazon Cognito como autorizador

Además de utilizar roles y políticas de IAM o autorizadores de Lambda (que anteriormente se denominaban autorizadores personalizados), también puede utilizar un grupo de usuarios de Amazon Cognito para controlar quién puede tener acceso a la API en Amazon API Gateway.

Para utilizar un grupo de usuarios de Amazon Cognito con la API, primero debe crear un autorizador del tipo `COGNITO_USER_POOLS` y configurar después un método de API para que utilice ese autorizador. Una vez implementada la API, lo primero que debe hacer el cliente es registrar al usuario en el grupo de usuarios, obtener un token de acceso o identidad para el usuario y llamar al método de API con uno de los tokens, que normalmente están establecidos en el encabezado `Authorization` de la solicitud. La llamada a la API solo tendrá éxito si se suministra el token solicitado y este es válido; de lo contrario, el cliente no tendrá autorización para hacer la llamada, ya que carecerá de las credenciales que permitirían autorizarlo.

El token de identidad se utiliza para autorizar las llamadas a la API en función de las solicitudes de identidad del usuario conectado. El token de acceso se utiliza para autorizar las llamadas a la API en función de los ámbitos personalizados de los recursos con protección de acceso especificados. Para obtener más información, consulte Uso de tokens con grupos de usuarios y Servidor de recursos y ámbitos personalizados.

Si desea crear y configurar un grupo de usuarios de Amazon Cognito para la API, debe realizar las siguientes tareas:

- Utilice la API, la CLI, el SDK o la consola de Amazon Cognito para crear un grupo de usuarios; también puede utilizar un grupo de usuarios que pertenezca a otra cuenta de AWS.
- Utilice la API, la CLI, el SDK o la consola de API Gateway para crear un autorizador de API Gateway con el grupo de usuarios elegido.
- Utilice la API, la CLI, el SDK o la consola de API Gateway para habilitar el autorizador de los métodos de API seleccionados.

Para llamar a los métodos de API con un grupo de usuarios habilitado, los clientes de la API tienen que realizar las siguientes tareas:

- Utilice la API, la CLI o el SDK de Amazon Cognito para registrar el usuario en el grupo de usuarios elegido y obtener un token de identidad o acceso. Para obtener más información sobre el uso de los SDK, consulte Ejemplos de código de Amazon Cognito con AWS SDK.
- Utilice una plataforma específica del cliente para llamar a la API de API Gateway implementada y proporcione el token apropiado en el encabezado `Authorization`.

Como desarrollador de la API, debe proporcionar a los desarrolladores del cliente el ID del grupo de usuarios, el ID del cliente y, probablemente, los secretos del cliente asociados que se hayan definido con el grupo de usuarios.

> **Note**
>
> Si desea permitir que un usuario inicie sesión con las credenciales de Amazon Cognito y pueda obtener también unas credenciales temporales para utilizarlas con los permisos de un rol de IAM, utilice las identidades federadas de Amazon Cognito. Para cada método HTTP de punto de enlace de recurso de API, establezca el tipo de autorización, categoría `Method Execution`, en `AWS_IAM`.

En esta sección, se explica cómo crear un grupo de usuarios, cómo integrar una API de API Gateway con el grupo de usuarios y cómo invocar una API que está integrada con el grupo de usuarios.

Temas

- Creación de un grupo de usuarios de Amazon Cognito para una API REST
- Integración de una API REST con un grupo de usuarios de Amazon Cognito
- Llamar a una API REST integrada con un grupo de usuarios de Amazon Cognito
- Configuración de un autorizador de Amazon Cognito entre cuentas para una API REST mediante la consola de API Gateway
- Cree un autorizador de Amazon Cognito para una API de REST con CloudFormation

Creación de un grupo de usuarios de Amazon Cognito para una API REST

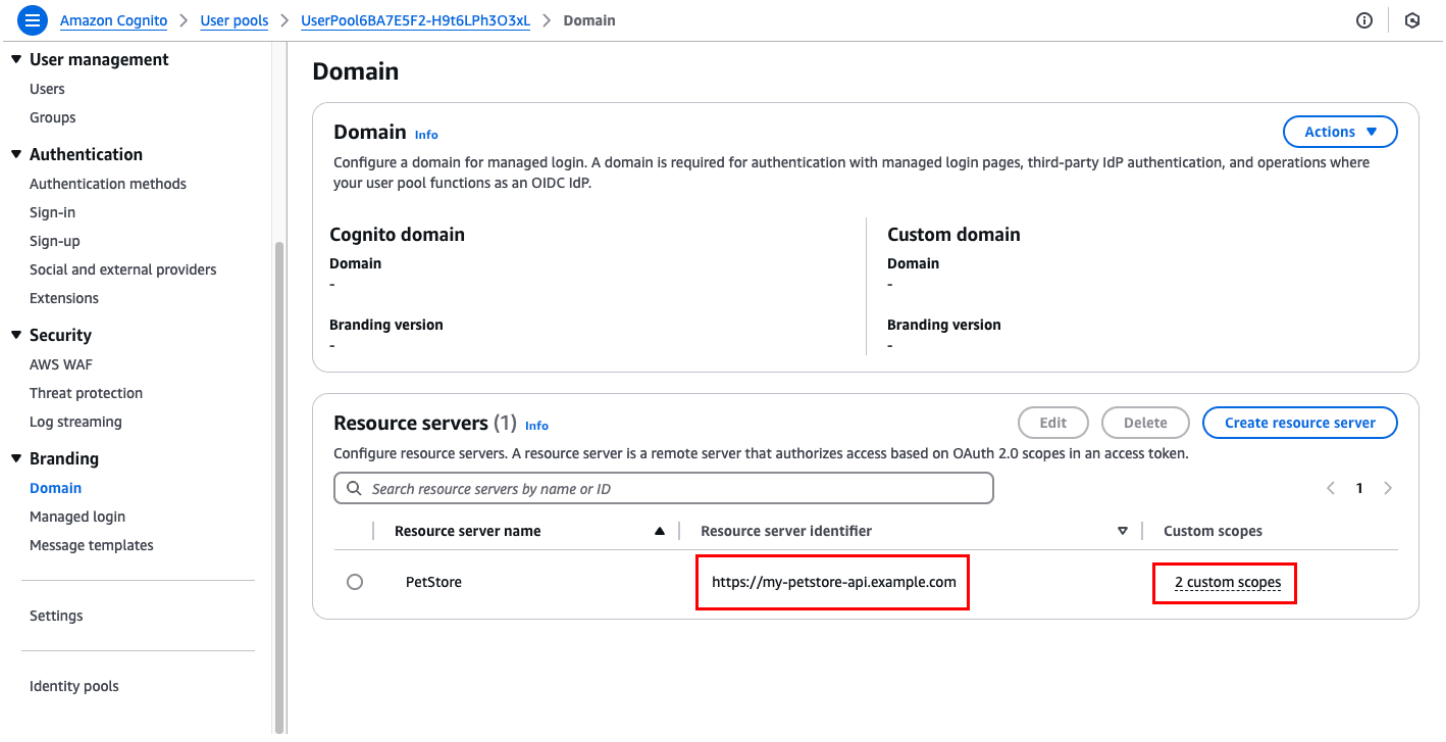
Antes de integrar la API con un grupo de usuarios, debe crear el grupo de usuarios en Amazon Cognito. La configuración del grupo de usuarios debe cumplir con todas las cuotas de recursos de Amazon Cognito. Todas las variables de Amazon Cognito definidas por el usuario, como grupos, usuarios y roles, deben usar solo caracteres alfanuméricos. Para obtener instrucciones sobre cómo

crear un grupo de usuarios, consulte Tutorial: Creación de un grupo de usuarios en la Guía para desarrolladores de Amazon Cognito.

Apunte el ID del grupo de usuarios, el ID del cliente y la clave secreta del cliente. El cliente debe proporcionarlos a Amazon Cognito para que el usuario pueda registrarse en el grupo de usuarios, para iniciar sesión en el grupo de usuarios y para obtener un token de identidad o acceso, que debe incluirse en las solicitudes para llamar a los métodos de la API configurados con el grupo de usuarios. Además, se debe especificar el nombre del grupo de usuarios al configurar el grupo de usuarios como autorizador de API Gateway, tal y como se describe a continuación.

Si utiliza tokens de acceso para autorizar las llamadas a los métodos de la API, asegúrese de configurar la integración de aplicaciones con el grupo de usuarios para establecer los ámbitos que desee en un determinado servidor de recursos. Para obtener más información sobre el uso de tokens con grupos de usuarios de Amazon Cognito, consulte Uso de tokens con grupos de usuarios. Para obtener más información acerca de los servidores de recursos, consulte Definir servidores de recursos para su grupo de usuarios.

Anote los identificadores del servidor de recursos configurados y los nombres de ámbito personalizados. Los necesita para generar los nombres completos de los ámbitos de acceso de OAuth Scopes (Ámbitos de OAuth) que el autorizador de `COGNITO_USER_POOLS` utiliza.



Integración de una API REST con un grupo de usuarios de Amazon Cognito

Después de crear un grupo de usuarios de Amazon Cognito, en API Gateway, debe crear un autorizador `COGNITO_USER_POOLS` que utilice el grupo de usuarios. El siguiente procedimiento muestra cómo hacer esto con la consola de API Gateway.

> **Note**
>
> Puede utilizar la acción `CreateAuthorizer` para crear un autorizador de `COGNITO_USER_POOLS` que utilice varios grupos de usuarios. Puede usar hasta 1000 grupos de usuarios para un autorizador de `COGNITO_USER_POOLS`. Este límite no se puede aumentar.

> **Important**
>
> Después de realizar cualquiera de los procedimientos que aparecen a continuación, deberá implementar o volver a implementar la API para propagar los cambios. Para obtener más información acerca cómo implementar una API, consulte Implementación de las API de REST en API Gateway.

Para crear un autorizador **`COGNITO_USER_POOLS`** mediante la consola de API Gateway

1. Cree una nueva API o seleccione una existente en API Gateway.
2. En el panel de navegación principal, elija Autorizadores.
3. Elija Crear autorizador.
4. Si desea configurar el nuevo autorizador para que utilice un grupo de usuarios, realice lo siguiente:

   a. En Nombre del autorizador, ingrese un nombre.
   b. En Tipo de autorizador, seleccione Cognito.
   c. En Grupo de usuarios de Cognito, elija la Región de AWS donde creó Amazon Cognito y seleccione un grupo de usuarios disponible.

      Puede utilizar una variable de etapa para definir su grupo de usuarios. Utilice el formato siguiente para su grupo de usuarios: `arn:aws:cognito-idp:`*`us-east-2`*`:`*`111122223333`*`:userpool/${stageVariables.`*`MyUserPool`*`}`.

d. En Origen del token, escriba **`Authorization`** como nombre del encabezado al que se va a pasar el token de identidad o acceso devuelto por Amazon Cognito cuando el usuario inicie sesión correctamente.

e. (Opcional) Ingrese una expresión regular en el campo Validación de token para validar el campo `aud` (audiencia) del token de identidad antes de autorizar la solicitud con Amazon Cognito. Tenga en cuenta que cuando se utiliza un token de acceso, esta validación rechaza la solicitud debido al token de acceso que no contiene el campo `aud`.

f. Elija Crear autorizador.

5. Después de crear el autorizador de `COGNITO_USER_POOLS`, puede realizar una prueba e invocarlo proporcionando un token de identidad aprovisionado desde el grupo de usuarios. No puede usar un token de acceso para realizar una prueba e invocar el autorizador.

   Puede obtener este token de identidad llamando al SDK de identidad de Amazon Cognito para realizar el inicio de sesión del usuario. También puede usar la acción `InitiateAuth`. Si no configura ningún Ámbito de autorización, API Gateway trata el token suministrado como un token de identidad.

El procedimiento anterior crea un autorizador `COGNITO_USER_POOLS` que utiliza el grupo de usuarios de Amazon Cognito que acaba de crearse. En función del modo en que se habilite el autorizador en un método de la API, puede utilizar un token de identidad o un token de acceso aprovisionado desde el grupo de usuarios integrados.

Para configurar un autorizador de **`COGNITO_USER_POOLS`** en los métodos

1. Seleccione Recursos. Elija un nuevo método o elija un método existente. Si es necesario, cree un recurso.
2. En la pestaña Solicitud de método, en Configuración de solicitud de método, elija Editar.
3. En Autorizador, en el menú desplegable, seleccione los autorizadores del grupo de usuarios de Amazon Cognito que acaba de crear.
4. Para utilizar un token de identidad, haga lo siguiente:

   a. Mantenga los ámbitos de autorización vacíos.

   b. Si fuera necesario, en la Solicitud de integración, agregue las expresiones `$context.authorizer.claims['`*`property-name`*`']` o `$context.authorizer.claims.`*`property-name`* a una plantilla de asignación de cuerpo para pasar la propiedad especificada de las reclamaciones de identidad desde

el grupo de usuarios al backend. En el caso de los nombres de propiedades sencillos, como `sub` o `custom-sub`, las dos notaciones son idénticas. En el caso de los nombres de propiedades complejos, como `custom:role`, no se puede usar la notación de puntos. Por ejemplo, las siguientes expresiones de asignación pasan los campos estándar `sub` y `email` de la reclamación al backend:

```
{
 "context" : {
  "sub" : "$context.authorizer.claims.sub",
  "email" : "$context.authorizer.claims.email"
 }
}
```

Si declaró un campo de reclamación personalizado al configurar un grupo de usuarios, puede seguir el mismo patrón para obtener acceso a los campos personalizados. El siguiente ejemplo obtiene un campo `role` personalizado de una reclamación:

```
{
 "context" : {
  "role" : "$context.authorizer.claims.role"
    }
}
```

Si el campo personalizado de la reclamación está declarado como `custom:role`, utilice el siguiente ejemplo para obtener la propiedad de la reclamación:

```
{
 "context" : {
  "role" : "$context.authorizer.claims['custom:role']"
    }
}
```

5. Para utilizar un token de acceso, haga lo siguiente:

   a. En Ámbitos de autorización, escriba uno o varios nombres completos de un ámbito que se haya configurado cuando se creó el grupo de usuarios de Amazon Cognito. Por ejemplo, siguiendo el ejemplo de Creación de un grupo de usuarios de Amazon Cognito para una API REST, uno de los ámbitos es `https://my-petstore-api.example.com/cats.read`.

En tiempo de ejecución, la llamada al método se realiza correctamente si el ámbito especificado en el método durante este paso coincide con el ámbito solicitado en el token de entrada. De lo contrario, la solicitud envía una respuesta de error `401 Unauthorized`.

b. Seleccione Save.

6. Repita estos pasos con otros métodos que elija.

Con el autorizador `COGNITO_USER_POOLS`, si la opción OAuth Scopes no está especificada, API Gateway trata el token proporcionado como un token de identidad y comprueba si la identidad solicitada se corresponde con alguna del grupo de usuarios. De lo contrario, API Gateway trata el token suministrado como un token de acceso y comprueba si los ámbitos de acceso solicitados en el token tienen coincidencias con los ámbitos de autorización declarados en el método.

En lugar de utilizar la consola de API Gateway, también puede habilitar un grupo de usuarios de Amazon Cognito en un método especificando un archivo de definición de OpenAPI e importando la definición de la API en API Gateway.

Para importar un autorizador COGNITO_USER_POOLS con un archivo de definición de OpenAPI

1. Cree (o exporte) un archivo de definición de OpenAPI para la API.
2. Especifique la definición JSON del autorizador `COGNITO_USER_POOLS` (`MyUserPool`) en la sección `securitySchemes` de OpenAPI 3.0 o en la sección `securityDefinitions` de Open API 2.0, tal como se indica a continuación:

   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. Para utilizar el token de identidad en la autorización del método, agregue `{ "MyUserPool": [] }` a la definición `security` del método, tal y como se muestra en el siguiente método GET del recurso raíz.

```
"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. Si desea utilizar el token de acceso para la autorización del método, cambie la definición de seguridad anterior a `{ "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. Si es necesario, puede establecer otra configuración de la API utilizando las definiciones o extensiones de OpenAPI correspondientes. Para obtener más información, consulte Extensiones de OpenAPI para API Gateway.

Llamar a una API REST integrada con un grupo de usuarios de Amazon Cognito

Para llamar a un método con un autorizador de grupo de usuarios configurado, el cliente debe hacer lo siguiente:

- Permitir al usuario inscribirse en el grupo de usuarios.
- Permitir al usuario iniciar sesión en el grupo de usuarios.
- Obtenga un token de identidad o acceso del usuario conectado del grupo de usuarios.
- Incluya el token en el encabezado `Authorization` (u otro encabezado que haya especificado al crear el autorizador).

Puede utilizar AWS Amplify para realizar estas tareas. Consulte Integrating Amazon Cognito With Web and Mobile Apps para obtener más información.

- Para Android, consulte Getting Started with Amplify for Android.
- Para utilizar iOS, consulte Getting started with Amplify for iOS.
- Para utilizar JavaScript, consulte Getting Started with Amplify for Javascript.

Configuración de un autorizador de Amazon Cognito entre cuentas para una API REST mediante la consola de API Gateway

Ahora también se puede utilizar un grupo de usuarios de Amazon Cognito perteneciente a otra cuenta de AWS como el autorizador de la API. El grupo de usuarios de Amazon Cognito puede utilizar estrategias de autenticación de token al portador como OAuth o SAML. Esto permite administrar y compartir fácilmente de forma centralizada un autorizador de grupo de usuarios de Amazon Cognito en distintas API de API Gateway.

En esta sección, mostramos cómo configurar un grupo de usuarios de Amazon Cognito entre cuentas mediante la consola de Amazon API Gateway.

En estas instrucciones, se presupone que usted ya dispone de una API de API Gateway en una cuenta de AWS y de un grupo de usuarios de Amazon Cognito en otra cuenta.

Creación de un autorizador de Amazon Cognito entre cuentas para una API de REST

Inicie sesión en la consola de Amazon API Gateway en la cuenta que tiene la API y, a continuación, haga lo siguiente:

1. Cree una nueva API o seleccione una existente en API Gateway.
2. En el panel de navegación principal, elija Autorizadores.
3. Elija Crear autorizador.
4. Si desea configurar el nuevo autorizador para que utilice un grupo de usuarios, realice lo siguiente:

    a. En Nombre del autorizador, ingrese un nombre.

    b. En Tipo de autorizador, seleccione Cognito.

    c. En Grupo de usuarios de Cognito, ingrese el ARN completo del grupo de usuarios que tiene en la segunda cuenta.

> **Note**
>
> En la consola de Amazon Cognito, puede encontrar el ARN de su grupo de usuarios en el campo Pool ARN (ARN de grupo) del panel General Settings (Configuración general).

d. En Origen del token, escriba **`Authorization`** como nombre del encabezado al que se va a pasar el token de identidad o acceso devuelto por Amazon Cognito cuando el usuario inicie sesión correctamente.

e. (Opcional) Ingrese una expresión regular en el campo Validación de token para validar el campo `aud` (audiencia) del token de identidad antes de autorizar la solicitud con Amazon Cognito. Tenga en cuenta que cuando se utiliza un token de acceso, esta validación rechaza la solicitud debido al token de acceso que no contiene el campo `aud`.

f. Elija Crear autorizador.

Cree un autorizador de Amazon Cognito para una API de REST con CloudFormation

Puede utilizar CloudFormation para crear un grupo de usuarios de Amazon Cognito y un autorizador de Amazon Cognito. En la plantilla de CloudFormation de ejemplo se realiza lo siguiente:

- Cree un grupo de usuarios de Amazon Cognito. El cliente primero registrar al usuario en el grupo de usuarios y obtener una identidad o un token de acceso. Si utiliza tokens de acceso para autorizar las llamadas a los métodos de la API, asegúrese de configurar la integración de aplicaciones con el grupo de usuarios para establecer los ámbitos que desee en un determinado servidor de recursos.
- Crea una API de API Gateway con un método `GET`.
- Crea un autorizador de Amazon Cognito que utiliza el encabezado `Authorization` como origen del token.

```
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
          - /
```

Integraciones para las API de REST en API Gateway

Una vez configurado un método de API, debe integrarlo con un punto de enlace en el backend. El punto de enlace del backend también se conoce como punto de enlace de integración y puede ser una función de Lambda, una página web HTTP o una acción de servicio de AWS.

Al igual que con el método de API, la integración de la API tiene una solicitud de integración y una respuesta de integración. La solicitud de integración incluye la solicitud HTTP que recibe el backend. Podría ser distinta o no de la solicitud de método enviada por el cliente. La respuesta de integración es la respuesta HTTP que encapsula la salida que devuelve el backend.

Configurar una solicitud de integración implica lo siguiente: configurar cómo transferir las solicitudes de método enviadas por el cliente al backend; configurar cómo transformar los datos de la solicitud, si fuese necesario, a los datos de la solicitud de integración; y especificar qué función de Lambda se debe llamar, especificando a qué servidor HTTP se debe reenviar la solicitud entrante o especificando la acción del servicio de AWS que se debe invocar.

Configurar una respuesta de integración (aplicable únicamente a integraciones que no sean de proxy) implica lo siguiente: configurar cómo transferir el resultado devuelto por el backend a una respuesta de método de un determinado código de estado; configurar cómo transformar los parámetros de respuesta de integración especificados en los parámetros de respuesta de método preconfigurados y configurar cómo asignar el cuerpo de la respuesta de integración al cuerpo de la respuesta de método con arreglo a las plantillas de mapeo de cuerpo especificadas.

Mediante programación, el recurso `Integration` encapsula la solicitud de integración y el recurso de API Gateway `IntegrationResponse`, la respuesta de integración.

Para configurar una solicitud de integración, debe crear un recurso `Integration` y utilizarlo para configurar la URL del punto de enlace de integración. A continuación, defina los permisos de IAM para acceder al backend y especifique los mapeos para transformar los datos de la solicitud entrante antes de transmitirlos al backend. Para configurar una respuesta de integración para una integración que no sea de proxy, debe crear un recurso `IntegrationResponse` y utilizarlo para establecer su respuesta de método de destino. A continuación, configure cómo asignar la salida del backend a la respuesta de método.

Temas

- Configuración de una solicitud de integración en API Gateway
- Configuración de una respuesta de integración en API Gateway
- Integraciones de Lambda para las API de REST en API Gateway
- Integraciones de HTTP para las API de REST en API Gateway
- Transmisión de la respuesta de integración para las integraciones de proxy en API Gateway
- Integraciones privadas para las API de REST en API Gateway
- Simulación de integraciones para las API de REST en API Gateway

Configuración de una solicitud de integración en API Gateway

Para configurar una solicitud de integración, debe llevar a cabo las siguientes tareas obligatorias y opcionales:

1. Elija un tipo de integración que determine cómo se transfieren los datos de la solicitud de método al backend.
2. Para las integraciones no simuladas, especifique un método HTTP y la URI del punto de enlace de integración específico, excepto para la integración `MOCK`.
3. Para las integraciones con funciones de Lambda y otras acciones de servicio de AWS, establezca un rol de IAM con los permisos necesarios para que API Gateway llame al backend en su nombre.
4. Para las integraciones que no sean de proxy, establezca el mapeo de parámetros necesario para asignar los parámetros de la solicitud de método predefinidos a los parámetros de la solicitud de integración adecuados.
5. Para las integraciones que no sean de proxy, establezca el mapeo de cuerpo necesario para asignar el cuerpo de la solicitud de método entrante de un determinado tipo de contenido con arreglo a la plantilla de mapeo especificada.
6. Para integraciones que no sean de proxy, especifique la condición bajo la cual los datos de la solicitud de método entrante se transfieren al backend tal y como están.
7. También puede especificar cómo administrar la conversión de tipo para una carga binaria.
8. También puede indicar un nombre del espacio de nombres de la caché y los parámetros de caché clave para habilitar el almacenamiento en caché de la API.

Estas tareas conllevan la creación de un recurso Integration de API Gateway y la definición de los valores adecuados para las propiedades. Puede hacerlo con la consola de API Gateway, los comandos de la AWS CLI, un AWS SDK o la API REST de API Gateway.

Temas

- Tareas básicas de una solicitud de integración de la API
- Elegir un tipo de integración de API de API Gateway
- Configuración de una integración de proxy con un recurso de proxy
- Configuración de una solicitud de integración de la API mediante la consola de API Gateway

Tareas básicas de una solicitud de integración de la API

Una solicitud de integración es una solicitud HTTP que envía API Gateway al backend, por la que se transfieren los datos de solicitud enviados por el cliente y se transforman los datos, si fuese necesario. El método HTTP (o verbo) y la URI de la solicitud de integración los dicta el backend (es decir, el punto de enlace de la integración). Pueden ser iguales o distintos al método HTTP y la URI de la solicitud de método, respectivamente.

Por ejemplo, cuando una función de Lambda devuelve un archivo que se recupera de Amazon S3, puede exponer esta operación de forma intuitiva como una solicitud de método `GET` al cliente, aunque la correspondiente solicitud de integración requiere que se utilice una solicitud `POST` para invocar la función de Lambda. En el caso de los puntos de enlace HTTP, es probable que tanto la solicitud del método como la correspondiente solicitud de integración utilicen el mismo verbo HTTP. Sin embargo, no es necesario. Puede integrar la siguiente solicitud de método:

```
GET /{var}?query=value
Host: api.domain.net
```

Con la siguiente solicitud de integración:

```
POST /
Host: service.domain.com
Content-Type: application/json
Content-Length: ...

{
   path: "{var}'s value",
   type: "value"
}
```

Como desarrollador de la API, puede utilizar el verbo HTTP y la URI para la solicitud de método que mejor se adapte a sus necesidades. Sin embargo, debe cumplir los requisitos del punto de enlace de integración. Cuando los datos de la solicitud de método son distintos de los datos de la solicitud de integración, puede solucionar esta diferencia asignando los datos de la solicitud de método a los datos de la solicitud de integración.

En los ejemplos anteriores, el mapeo traduce los valores de la variable de ruta (`{var}`) y el parámetro de consulta (`query`) de la solicitud de método `GET` en los valores de las propiedades de carga de la solicitud de integración para `path` y `type`. Podrían asignarse otros datos de la

solicitud, como los encabezados y el cuerpo. Esto se describe en the section called “Asignación de parámetros”.

Al configurar la solicitud HTTP o de integración de proxy HTTP, asigne la URL del punto de enlace HTTP del backend como el valor de la URI de la solicitud de integración. Por ejemplo, en la API de PetStore, la solicitud de método para obtener una página de mascotas tiene la siguiente URI de la solicitud de integración:

```
http://petstore-demo-endpoint.execute-api.com/petstore/pets
```

Al configurar la integración de Lambda o de proxy de Lambda asigne el nombre de recurso de Amazon (ARN) para invocar la función de Lambda como el valor de la URI de la solicitud de integración. El ARN tiene el siguiente formato:

```
arn:aws:apigateway:api-region:lambda:path//2015-03-31/functions/arn:aws:lambda:lambda-region:account-id:function:lambda-function-name/invocations
```

La parte que aparece después de `arn:aws:apigateway:`*`api-region`*`:lambda:path/`, es decir, `/2015-03-31/functions/arn:aws:lambda:`*`lambda-region`*`:`*`account-id`*`:function:`*`lambda-function-name`*`/invocations`, corresponde a la ruta de URI de la API REST de la acción de Lambda Invoke. Si utiliza la consola de API Gateway para configurar la integración de Lambda, API Gateway crea el ARN y lo asigna a la URI de integración después de que se le solicite que elija el *`lambda-function-name`* de una región.

Cuando se configura la solicitud de integración con otra acción de servicio de AWS, la URI de la solicitud de integración también es un ARN, de modo similar a la integración con la acción `Invoke` de Lambda. Por ejemplo, en el caso de la integración con la acción GetBucket de Amazon S3, la URI de la solicitud de integración es un ARN con el siguiente formato:

```
arn:aws:apigateway:api-region:s3:path/{bucket}
```

La URI de la solicitud de integración utiliza la convención de ruta para especificar la acción, donde *`{bucket}`* es el marcador de posición de un nombre de bucket. También se puede hacer referencia a una acción del servicio de AWS por su nombre. Si se utiliza el nombre de la acción, la URI de la solicitud de integración para la acción `GetBucket` de Amazon S3 se convierte en lo siguiente:

```
arn:aws:apigateway:api-region:s3:action/GetBucket
```

Con la URI de la solicitud de integración basada en la acción, el nombre del bucket (*{bucket}*) se debe especificar en el cuerpo de la solicitud de integración (`{ Bucket: "{bucket}" }`), siguiendo el formato de entrada de la acción `GetBucket`.

Para las integraciones de AWS, también debe configurar las credenciales para permitir que API Gateway llame a las acciones integradas. Puede crear uno nuevo o bien elegir un rol de IAM existente para que API Gateway llame a la acción y, a continuación, especificar el rol mediante su ARN. A continuación se muestra un ejemplo de este ARN:

```
arn:aws:iam::account-id:role/iam-role-name
```

Este rol de IAM debe contener una política para permitir que se ejecute la acción. También se debe haber declarado API Gateway (en la relación de confianza del rol) como una entidad de confianza para asumir el rol. Estos permisos se pueden conceder en la propia acción. Se conocen como permisos basados en recursos. Para la integración de Lambda, puede llamar a la acción addPermission de Lambda para definir los permisos basados en recursos y, a continuación, establecer `credentials` en null en la solicitud de integración de API Gateway.

Hemos analizado los aspectos básicos de la configuración de la integración. La configuración avanzada implica la asignación de los datos de la solicitud de método a los datos de la solicitud de integración. Para obtener más información, consulte the section called “Transformaciones de datos”.

Elegir un tipo de integración de API de API Gateway

El tipo de integración de la API se selecciona con arreglo a los tipos de punto de enlace de integración con los que trabaje y al modo en que desea transferir los datos hacia y desde el punto de enlace de integración. Para una función de Lambda, puede tener la integración de proxy de Lambda o la integración personalizada de Lambda. Para un punto de enlace HTTP, puede elegir entre la integración de proxy HTTP o una integración HTTP personalizada. Para una acción del servicio de AWS, dispone de la integración de AWS solo para el tipo que no sea proxy. API Gateway también admite la integración simulada, donde API Gateway actúa como punto de enlace de integración para responder a una solicitud de método.

La integración personalizada de Lambda es un caso especial de la integración de AWS, donde el punto de enlace de integración se corresponde con la acción que invoca las funciones del servicio de Lambda.

Mediante programación, puede seleccionar un tipo de integración configurando la propiedad `type` en el recurso `Integration`. Para la integración de proxy de Lambda, el valor es `AWS_PROXY`.

Para la integración personalizada de Lambda y el resto de las integraciones de AWS, el valor es `AWS`. Para la integración de proxy HTTP y la integración HTTP, el valor es `HTTP_PROXY` y `HTTP`, respectivamente. Para la integración simulada, el valor `type` es `MOCK`.

La integración de proxy de Lambda admite una configuración de integración simplificada con una única función de Lambda. La configuración es sencilla y puede evolucionar con el backend sin tener que eliminar la configuración existente. Por estas razones, es muy recomendable para la integración con una función de Lambda.

En cambio, la integración de Lambda personalizada permite la reutilización de las plantillas de mapeo configuradas para varios puntos de enlace de integración con requisitos similares sobre el formato de los datos de entrada y salida. Esta configuración es más compleja y se recomienda para situaciones de aplicación más avanzadas.

Del mismo modo, la integración de proxy HTTP tiene una configuración de integración simplificada y puede evolucionar con el backend sin tener que eliminar la configuración existente. La integración HTTP personalizada resulta más compleja de configurar, pero permite la reutilización de las plantillas de mapeo configuradas para otros puntos de enlace de integración.

En la siguiente lista se resumen los tipos de integración admitidos:

- `AWS`: este tipo de integración permite a una API exponer acciones del servicio de AWS. En la integración de `AWS`, debe configurar tanto la solicitud de integración como la respuesta de integración y también configurar el mapeo necesario de los datos entre la solicitud de método y la solicitud de integración, y entre la respuesta de integración y la respuesta de método.
- `AWS_PROXY`: este tipo de integración permite que se integre un método de la API con la acción de invocación de la función de Lambda a través de una configuración de integración flexible, versátil y simplificada. Esta integración se basa en las interacciones directas entre el cliente y la función de Lambda integrada.

  Con este tipo de integración, que también se conoce como integración de proxy de Lambda, no tiene que establecer la solicitud de integración ni la respuesta de integración. API Gateway transmite la solicitud entrante del cliente, como entrada, a la función de Lambda del backend. La función de Lambda integrada toma la entrada de este formato y analiza la entrada de todos los orígenes disponibles, incluidos los encabezados de solicitudes, las variables de ruta de la URL, los parámetros de cadena de consulta y el cuerpo aplicable. Esta función devuelve un resultado con el siguiente formato de salida.

Este es el tipo de integración recomendable para llamar a la función de Lambda a través de API Gateway y no es aplicable a ninguna otra acción de servicio de AWS, incluidas las acciones de Lambda que no sean la acción que invoca las funciones.

- `HTTP`: este tipo de integración permite que una API exponga puntos de enlace de HTTP en el backend. Con la integración `HTTP`, que también se conoce como integración HTTP personalizada, debe configurar tanto la solicitud de integración como la respuesta de integración. Debe establecer la asignación necesaria de los datos entre la solicitud de método y la solicitud de integración, y entre la respuesta de integración y la respuesta de método.
- `HTTP_PROXY`: La integración de proxy HTTP permite que un cliente tenga acceso a puntos de enlace HTTP de backend con una configuración de la integración simplificada en un único método de API. No tiene que establecer la solicitud de integración ni la respuesta de integración. API Gateway pasa la solicitud entrante del cliente al punto de enlace HTTP y pasa la respuesta saliente del punto de enlace HTTP al cliente.
- `MOCK`: este tipo de integración permite que API Gateway devuelva una respuesta sin enviar la solicitud al backend. Esto es útil en las pruebas de la API, ya que se puede utilizar para probar la configuración de la integración sin incurrir en cargos por utilizar el backend y para permitir el desarrollo colaborativo de una API.

  En el desarrollo colaborativo, un equipo puede aislar sus esfuerzos de desarrollo configurando simulaciones de los componentes de la API que son propiedad de otros equipos mediante el uso de integraciones `MOCK`. También se utiliza para devolver encabezados relacionados con CORS y garantizar que el método de API permita el acceso a CORS. De hecho, la consola de API Gateway integra el método `OPTIONS` para admitir CORS con una integración simulada. Las respuestas de gateway son otros ejemplos de integraciones simuladas.

Configuración de una integración de proxy con un recurso de proxy

Para configurar una integración de proxy en una API de API Gateway con un recurso de proxy, realice las tareas siguientes:

- Crear un recurso de proxy con la variable de ruta expansiv `{proxy+}`.
- Establecer el método `ANY` en el recurso de proxy
- Integrar el recurso y el método con un backend mediante el tipo de integración HTTP o Lambda.

> **Note**
>
> Las variables de ruta expansiva, los métodos ANY y los tipos de integración de proxy son características independientes, aunque se utilizan normalmente juntas. Puede configurar un método HTTP específico en un recurso expansivo o aplicar tipos de integración distintos de proxy en un recurso de proxy.

API Gateway establece determinadas restricciones y limitaciones al manipular métodos con una integración de proxy de Lambda o una integración de proxy HTTP. Para obtener información, consulte the section called “Notas importantes”.

> **Note**
>
> Cuando se utiliza la integración de proxy con acceso directo, API Gateway devuelve el encabezado predeterminado `Content-Type:application/json` si no se especifica el tipo de contenido de una carga.

Un recurso de proxy es más eficaz cuando se integra con un backend que usa la integración de proxy HTTP o la integración de proxy de Lambda.

Integración de proxy HTTP con un recurso de proxy

La integración de proxy HTTP, designada por `HTTP_PROXY` en la API REST de API Gateway, es para la integración de una solicitud de método con un punto de enlace HTTP del backend. Con este tipo de integración, API Gateway simplemente transmite toda la solicitud y la respuesta entre el frontend y el backend, aunque esto está sujeto a determinadas restricciones y limitaciones.

> **Note**
>
> La integración de proxy HTTP admite encabezados y cadenas de consulta con varios valores.

Cuando aplica la integración de proxy HTTP a un recurso de proxy, puede configurar la API para que exponga una parte o la totalidad de la jerarquía del punto de enlace del backend HTTP con una sola integración configurada. Supongamos, por ejemplo, que el backend del sitio web está organizado en

varias ramas de nodos de árbol desde el nodo raíz (`/site`) como: `/site/`a_0`/`a_1`/.../`a_N, `/site/`b_0`/`b_1`/.../`b_M, etc. Si integra el método `ANY` en un recurso de proxy de `/api/{proxy+}` con los puntos de enlace del backend con rutas URL de `/site/{proxy}`, una sola solicitud de integración puede admitir todas las operaciones HTTP (GET, POST, etc.) en cualquiera de [a_0, a_1, ..., a_N, b_0, b_1, ...b_M, ...]. Si aplica una integración de proxy a un método HTTP específico (por ejemplo, `GET`), en su lugar, la solicitud de integración resultante funcionará con las operaciones especificadas (es decir, `GET`) en cualquiera de los nodos del backend.

Integración de proxy de Lambda con un recurso de proxy

La integración de proxy de Lambda, designada por `AWS_PROXY` en la API REST de API Gateway, es para la integración de una solicitud de método con una función de Lambda en el backend. Con este tipo de integración, API Gateway aplica una plantilla de mapeo predeterminada para enviar toda la solicitud a la función de Lambda y transforma la salida de la función de Lambda en respuestas HTTP.

Del mismo modo, puede aplicar la integración de proxy de Lambda a un recurso proxy de `/api/{proxy+}` para configurar una sola integración, si desea que la función de Lambda del backend reaccione individualmente a los cambios que se produzcan en cualquiera de los recursos de la API bajo `/api`.

Configuración de una solicitud de integración de la API mediante la consola de API Gateway

La configuración de un método de API define el método y describe sus comportamientos. Para configurar un método, debe especificar un recurso, incluida la raíz ("/"), en el que se expone el método, un método HTTP (`GET`, `POST`, etc.) y la forma en que se integrará con el backend de destino. La solicitud y la respuesta del método especifican el contrato con la aplicación que realiza la llamada, que estipula los parámetros que puede recibir la API y el aspecto que tendrá la respuesta.

Los siguientes procedimientos describen cómo utilizar la consola de API Gateway para crear una solicitud de integración.

Temas

- Configuración de una integración de Lambda
- Configuración de integración de HTTP
- Configuración de una integración servicios de AWS
- Configuración de una integración simulada

Configuración de una integración de Lambda

Utilice una integración de función de Lambda para integrar la API con una función de Lambda. A nivel de API, es un tipo de integración de `AWS` si se crea una integración sin proxy o un tipo de integración de `AWS_PROXY` si se crea una integración de proxy.

Configuración de una integración de Lambda

1. En el panel Recursos, elija Crear método.
2. En Tipo de método, seleccione un método HTTP.
3. En Tipo de integración, seleccione Función Lambda.
4. Para usar una integración de proxy de Lambda, active la Integración de proxy de Lambda. Para obtener más información sobre las integraciones de proxy de Lambda, consulte the section called “ Descripción de la integración de proxy de Lambda ”.
5. En Función de Lambda, ingrese el nombre de la función de Lambda.

   Si utiliza una función de Lambda en una región diferente a la de su API, seleccione la región en el menú desplegable e ingrese el nombre de la función de Lambda. Si utiliza una función de Lambda entre cuentas, ingrese el ARN de la función.
6. Si desea utilizar el valor predeterminado del tiempo de espera, que es de 29 segundos, mantenga activado el Tiempo de espera predeterminado. Para establecer un tiempo de espera personalizado, elija Tiempo de espera predeterminado e ingrese un valor de tiempo de espera comprendido entre `50` y `29000` milisegundos.
7. (Opcional) Puede configurar los ajustes de solicitud de método mediante los siguientes menús desplegables. Elija los Ajustes de solicitud de método y configure la solicitud de método. Para obtener más información, consulte el paso 3 de the section called “Edición de una solicitud de método en la consola”.

   También puede configurar los ajustes de solicitud de método después de crear el método.
8. Elija Crear método.

Configuración de integración de HTTP

Utilice una integración de HTTP para integrar la API con un punto de conexión HTTP. En el nivel de API, este es el tipo de integración `HTTP`.

Configuración de integración de HTTP

1. En el panel Recursos, elija Crear método.
2. En Tipo de método, seleccione un método HTTP.
3. Para Tipo de integración, elija HTTP.
4. Para usar una integración de proxy HTTP, active la Integración de proxy HTTP. Para obtener más información acerca de las integraciones de proxy HTTP, consulte the section called “Configurar integraciones de proxy HTTP en API Gateway”.
5. En HTTP method (Método HTTP), elija el tipo de método HTTP que más se parezca al método del backend HTTP.
6. En URL del punto de conexión, ingrese la dirección URL del backend HTTP que desea que utilice este método.
7. En Tratamiento de contenido, seleccione un comportamiento de tratamiento del contenido.
8. Si desea utilizar el valor predeterminado del tiempo de espera, que es de 29 segundos, mantenga activado el Tiempo de espera predeterminado. Para establecer un tiempo de espera personalizado, elija Tiempo de espera predeterminado e ingrese un valor de tiempo de espera comprendido entre `50` y `29000` milisegundos.
9. (Opcional) Puede configurar los ajustes de solicitud de método mediante los siguientes menús desplegables. Elija los Ajustes de solicitud de método y configure la solicitud de método. Para obtener más información, consulte el paso 3 de the section called “Edición de una solicitud de método en la consola”.

   También puede configurar los ajustes de solicitud de método después de crear el método.
10. Elija Crear método.

Configuración de una integración servicios de AWS

Utilice una integración de servicios de AWS para integrar la API directamente con un servicio de AWS. En el nivel de API, este es el tipo de integración `AWS`.

Para configura una API de API Gateway, puede hacer lo siguiente:

- Crear una nueva función de Lambda.
- Configurar un permiso de recurso en la función de Lambda.
- Realizar cualquier otra acción del servicio Lambda.

Debe elegir el Servicio de AWS.

Configuración de una integración de servicios de AWS

1. En el panel Recursos, elija Crear método.
2. En Tipo de método, seleccione un método HTTP.
3. En Tipo de integración, seleccione Servicio de AWS.
4. En AWS Region, elija la región de AWS que desea que utilice este método para llamar a la acción.
5. En Servicio de AWS, elija el servicio de AWS al que desea que llame este método.
6. En Subdominio de AWS, ingrese el subdominio que utiliza el servicio de AWS. Normalmente, puede dejarlo en blanco. Algunos servicios de AWS admiten subdominios como parte de los hosts. En la documentación del servicio encontrará la disponibilidad y los detalles, si están disponibles.
7. En HTTP method (Método HTTP), seleccione el tipo de método HTTP correspondiente a la acción. Para el tipo de método HTTP, consulte la documentación de referencia de la API correspondiente al servicio de AWS que eligió en Servicio de AWS.
8. En Tipo de acción, seleccione Usar nombre de acción para usar una acción de la API o Usar sustitución de ruta para usar una ruta de recursos personalizada. Para conocer las acciones disponibles y las rutas de recurso personalizadas, consulte la documentación de referencia de la API correspondiente al servicio de AWS que eligió en Servicio de AWS.
9. Ingrese un Nombre de acción o una Sustitución de ruta.
10. En Rol de ejecución, ingrese el ARN del rol de IAM que usará el método para llamar a la acción.

    Para crear un rol de IAM, puede adaptar las instrucciones de the section called “Paso 1: Crear el rol de ejecución del proxy de servicio de AWS”. Especifique una política de acceso con el número de acciones e instrucciones de recursos que desee. Para obtener más información, consulte the section called “Cómo funciona Amazon API Gateway con IAM”.

    Para ver la sintaxis de las acciones e instrucciones de recursos, consulte la documentación del servicio de AWS que eligió en Servicio de AWS.

    Para la relación de confianza del rol de IAM, especifique lo siguiente, que permite a API Gateway realizar acciones en nombre de su cuenta de AWS:

JSON

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

11. Si desea utilizar el valor predeterminado del tiempo de espera, que es de 29 segundos, mantenga activado el Tiempo de espera predeterminado. Para establecer un tiempo de espera personalizado, elija Tiempo de espera predeterminado e ingrese un valor de tiempo de espera comprendido entre `50` y `29000` milisegundos.
12. (Opcional) Puede configurar los ajustes de solicitud de método mediante los siguientes menús desplegables. Elija los Ajustes de solicitud de método y configure la solicitud de método. Para obtener más información, consulte el paso 3 de the section called “Edición de una solicitud de método en la consola”.

    También puede configurar los ajustes de solicitud de método después de crear el método.
13. Elija Crear método.

Configuración de una integración simulada

Use una integración simulada si desea que API Gateway actúe como backend y devuelva respuestas estáticas. En el nivel de API, este es el tipo de integración `MOCK`. Normalmente, puede usar la integración `MOCK` si la API aún no es definitiva, pero desea generar respuestas de API para permitir que los equipos dependientes realicen pruebas. En el método `OPTION`, API Gateway establece la integración `MOCK` como predeterminada para devolver encabezados que habilitan CORS para el recurso de la API que se aplicó.

Configuración de una integración simulada

1. En el panel Recursos, elija Crear método.
2. En Tipo de método, seleccione un método HTTP.
3. Para Tipo de integración, elija Simulación.
4. (Opcional) Puede configurar los ajustes de solicitud de método mediante los siguientes menús desplegables. Elija los Ajustes de solicitud de método y configure la solicitud de método. Para obtener más información, consulte el paso 3 de the section called “Edición de una solicitud de método en la consola”.

   También puede configurar los ajustes de solicitud de método después de crear el método.
5. Elija Crear método.

Configuración de una respuesta de integración en API Gateway

Para una integración que no sea de proxy, debe configurar al menos una respuesta de integración, y convertirla en la respuesta predeterminada, para pasar el resultado que devuelve el backend al cliente. Puede optar por pasar el resultado como está o transformar los datos de la respuesta de integración en los datos de la respuesta de método si ambos tienen formatos diferentes.

Para una integración de proxy, API Gateway pasa automáticamente la salida del backend al cliente como una respuesta HTTP. No debe configurar la respuesta de integración ni la respuesta del método.

Para configurar una respuesta de integración, debe llevar a cabo las siguientes tareas obligatorias y opcionales:

1. Especifique el código de estado HTTP de la respuesta del método a la que se asignan los datos de la respuesta de integración. Esto es necesario.
2. Defina una expresión regular para seleccionar que la salida del backend esté representada por esta respuesta de integración. Si lo deja vacío, la respuesta será la respuesta predeterminada que se utiliza para detectar las respuestas que no se hayan configurado aún.
3. Si es necesario, declare mapeos compuestos de pares de clave-valor para asignar los parámetros de la respuesta de integración especificados a los parámetros de una respuesta de método determinada.
4. Si es necesario, añada plantillas de mapeo de cuerpo para transformar las cargas de una respuesta de integración determinada en las cargas de la respuesta de método especificada.

5. Si es necesario, especifique cómo administrar la conversión de tipo para una carga binaria.

Una respuesta de integración es una respuesta HTTP que encapsula la respuesta del backend. Para un punto de enlace HTTP, la respuesta del backend es una respuesta HTTP. El código de estado de la respuesta de integración puede tomar el código de estado que devuelve el backend, y el cuerpo de la respuesta de integración es la carga que devuelve el backend. Para un punto de enlace Lambda, la respuesta del backend es la salida que devuelve la función de Lambda. Con la integración de Lambda, la salida de la función de Lambda se devuelve como una respuesta `200 OK`. La carga puede contener el resultado como datos JSON, incluida una cadena JSON o un objeto JSON, o un mensaje de error como un objeto JSON. Puede asignar una expresión regular a la propiedad selectionPattern para asignar una respuesta de error a la respuesta de error HTTP adecuada. Para obtener más información acerca de la respuesta de error de una función de Lambda, consulte Gestionar los errores de Lambda en API Gateway. Con la integración de proxy de Lambda, la función de Lambda debe devolver una salida con el siguiente formato:

```
{
    statusCode: "...",            // a valid HTTP status code
    headers: {
        custom-header: "..."      // any API-specific custom header
    },
    body: "...",                  // a JSON string.
    isBase64Encoded:  true|false  // for binary support
}
```

No es necesario asignar la respuesta de la función de Lambda a su respuesta HTTP adecuada.

Para devolver el resultado al cliente, configure la respuesta de integración para pasar la respuesta del punto de enlace como está a la correspondiente respuesta de método. También puede asignar los datos de la respuesta del punto de enlace a los datos de la respuesta de método. Entre los datos de respuesta que se pueden asignar se incluyen el código de estado de la respuesta, los parámetros de encabezado de la respuesta y el cuerpo de la respuesta. Si no se define ninguna respuesta de método para el código de estado devuelto, API Gateway devuelve un error 500. Para obtener más información, consulte Anulación de los parámetros de solicitud y respuesta y de los códigos de estado de la API para las API de REST en API Gateway.

Integraciones de Lambda para las API de REST en API Gateway

Puede integrar un método de API con una función de Lambda mediante la integración de proxy de Lambda o la integración de Lambda que no sea de proxy (personalizada).

En la integración de proxy de Lambda, la configuración requerida es sencilla. Establezca el método HTTP de la integración en POST, la URI del punto de conexión de la integración en el ARN de la acción de invocación de una función de Lambda específica y otorgue permisos a API Gateway para llamar a la función de Lambda en su nombre.

En la integración de Lambda que no sea de proxy, además de los pasos de configuración de integración de proxy, también debe especificar cómo se asignan los datos entrantes a la solicitud de integración y cómo se asignan los datos resultantes de respuesta de la integración a la respuesta de método.

Temas

- Integraciones de proxy de Lambda en API Gateway
- Configuración de integraciones de Lambda personalizadas en API Gateway
- Configurar la invocación asíncrona de la función de Lambda de backend
- Gestionar los errores de Lambda en API Gateway

Integraciones de proxy de Lambda en API Gateway

En la siguiente sección se muestra cómo utilizar una integración de proxy de Lambda.

Temas

- Descripción de la integración de proxy de Lambda en API Gateway
- Compatibilidad con encabezados de varios valores y parámetros de cadenas de consulta
- Formato de entrada de una función de Lambda para la integración de proxy
- Formato de salida de una función de Lambda para la integración de proxy
- Configuración de una integración de proxy de Lambda para API Gateway mediante la AWS CLI
- Configuración de un recurso de proxy con la integración de proxy de Lambda con una definición OpenAPI

Descripción de la integración de proxy de Lambda en API Gateway

La integración de proxy de Lambda de Amazon API Gateway es un mecanismo sencillo, potente y ágil para desarrollar una API con la configuración de un solo método de API. La integración de proxy de Lambda permite al cliente llamar a una única función de Lambda en el backend. La función obtiene acceso a muchos recursos o características de otros servicios de AWS, incluidas las llamadas a otras funciones de Lambda.

En la integración de proxy de Lambda, cuando un cliente envía una solicitud de API, API Gateway pasa a la función de Lambda integrada un evento de objeto, salvo que no se conserve el orden de los parámetros de la solicitud. Estos datos de la solicitud incluyen los encabezados de solicitud, los parámetros de cadena de consulta, las variables de ruta de la URL, la carga y los datos de configuración de la API. Los datos de configuración pueden incluir el nombre de la etapa de implementación actual, las variables de la etapa, la identidad del usuario o el contexto de autorización (si lo hubiera). La función de Lambda del backend analiza los datos de la solicitud entrante para determinar la respuesta que devuelve. Para que API Gateway transmita la salida de Lambda como la respuesta de la API al cliente, la función de Lambda debe devolver el resultado en este formato.

Dado que API Gateway no interviene mucho entre el cliente y la función de Lambda del backend para la integración de proxy de Lambda, el cliente y la función de Lambda integrada pueden adaptarse a los cambios de cada uno sin interrumpir la configuración de integración existente de la API. Para habilitar esto, el cliente debe seguir los protocolos de aplicación promulgados por la función de Lambda del backend.

Puede configurar una integración de proxy de Lambda para cualquier método de API. Sin embargo, una integración de proxy de Lambda es más potente cuando se configura para un método de API que implica un recurso de proxy genérico. El recurso de proxy genérico se puede indicar mediante una variable especial de ruta basada en plantilla de `{proxy+}`, el marcador de posición del método catch-all `ANY` o ambos. El cliente puede transmitir la entrada a la función de Lambda del backend en la solicitud entrante según lo soliciten los parámetros o la carga aplicable. Los parámetros de solicitud incluyen encabezados, variables de ruta de la URL, parámetros de cadenas de consulta y la carga aplicable. La función de Lambda integrada verifica todas las fuentes de entrada antes de procesar la solicitud y de responder al cliente con mensajes de error importantes si falta alguna de las entradas requeridas.

Al llamar a un método de API integrado con el método HTTP genérico de `ANY` y el recurso genérico de `{proxy+}`, el cliente envía una solicitud con un método HTTP específico en lugar de `ANY`. El

cliente también especifica una ruta de URL específica en lugar de `{proxy+}` e incluye cualquier encabezado, parámetro de cadena de consulta o una carga aplicable.

En la siguiente lista se resumen los comportamientos de tiempo de ejecución de los diferentes métodos de API con la integración de proxy de Lambda:

- `ANY /{proxy+}`: el cliente debe elegir un método HTTP específico, debe establecer una jerarquía de ruta de recursos específica y puede establecer cualquier encabezado, parámetro de cadena de consulta y carga aplicable para transmitir los datos como entrada a la función de Lambda integrada.
- `ANY /res`: el cliente debe elegir un método HTTP específico y puede establecer cualquier encabezado, parámetro de cadena de consulta y carga aplicable para transmitir los datos como entrada a la función de Lambda integrada.
- `GET|POST|PUT|... /{proxy+}`: el cliente puede establecer una jerarquía de ruta de recursos específica, cualquier encabezado, parámetro de cadena de consulta y carga aplicable para transmitir los datos como entrada a la función de Lambda integrada.
- `GET|POST|PUT|... /res/{path}/...`: el cliente debe elegir un segmento de ruta específico (para la variable `{path}`) y puede establecer cualquier encabezado de solicitud, parámetro de cadena de consulta y carga aplicable para transmitir los datos de entrada a la función de Lambda integrada.
- `GET|POST|PUT|... /res`: el cliente puede elegir cualquier encabezado de solicitud, parámetro de cadena de consulta y carga aplicable para transmitir los datos de entrada a la función de Lambda integrada.

Tanto el recurso de proxy de `{proxy+}` como el recurso personalizado de `{custom}` se expresan como variables de ruta basada en plantilla. Sin embargo, `{proxy+}` puede hacer referencia a cualquier recurso en una jerarquía de ruta, mientras que `{custom}` se refiere únicamente a un segmento de ruta específico. Por ejemplo, una tienda de comestibles podría organizar un inventario de productos en línea por nombres de departamento, categorías de productos y tipos de productos. El sitio web de la tienda de comestibles puede entonces representar los productos disponibles con las siguientes variables de ruta basada en plantilla para los recursos personalizados: `/{department}/{produce-category}/{product-type}`. Por ejemplo, las manzanas están representadas con `/produce/fruit/apple` y las zanahorias con `/produce/vegetables/carrot`. También puede utilizar `/{proxy+}` para representar a cualquier departamento, cualquier categoría de producto o cualquier tipo de producto que un cliente puede buscar mientras realiza sus

compras en la tienda en línea. Por ejemplo, `/{proxy+}` puede hacer referencia a cualquiera de los siguientes elementos:

- `/produce`
- `/produce/fruit`
- `/produce/vegetables/carrot`

Para que los clientes puedan buscar cualquier producto disponible, su categoría de producto y el departamento asociado en la tienda, puede exponer un único método de `GET /{proxy+}` con permisos de solo lectura. Del mismo modo, para permitir que un supervisor actualice el inventario del departamento `produce`, puede configurar otro método único de `PUT /produce/{proxy+}` con permisos de lectura y escritura. Para permitir que un cajero actualice el total de unidades de una verdura, puede configurar un método `POST /produce/vegetables/{proxy+}` con permisos de lectura y escritura. Para que el gerente de una tienda pueda realizar cualquier acción posible en cualquier producto disponible, el desarrollador de la tienda en línea puede exponer el método `ANY /{proxy+}` con permisos de lectura y escritura. En cualquier caso, en el tiempo de ejecución, el cliente o el empleado debe seleccionar un producto específico de un tipo determinado en un departamento elegido o en un departamento específico.

Para obtener más información acerca de la configuración de las integraciones del proxy de API Gateway, consulte Configuración de una integración de proxy con un recurso de proxy.

La integración del proxy requiere que el cliente tenga un conocimiento más detallado de los requisitos del backend. Por lo tanto, para garantizar un óptimo rendimiento de la aplicación y experiencia del usuario, el desarrollador del backend debe comunicar con claridad al desarrollador del cliente los requisitos del backend y proporcionar un mecanismo de comentarios de errores sólido cuando los requisitos no se cumplan.

Compatibilidad con encabezados de varios valores y parámetros de cadenas de consulta

API Gateway ahora es compatible con varios encabezados y parámetros de cadena de consulta que tengan el mismo nombre. Los encabezados de varios valores y los encabezados y parámetros de un solo valor se pueden combinar en las mismas solicitudes y respuestas. Para obtener más información, consulte Formato de entrada de una función de Lambda para la integración de proxy y Formato de salida de una función de Lambda para la integración de proxy.

Formato de entrada de una función de Lambda para la integración de proxy

Con la integración de proxy de Lambda, API Gateway asigna toda la solicitud de cliente al parámetro de entrada `event` de la función de Lambda del backend. En el siguiente ejemplo, se muestra la estructura de un evento que API Gateway envía a una integración de proxy de Lambda.

En este ejemplo, suponemos que la invocación a API Gateway fue la siguiente:

```
curl 'https://a1b2c3.execute-api.us-east-1.amazonaws.com/my/path?
parameter1=value1&parameter2=value1&parameter2=value2&parameter3=value1,value2'
 -H 'header1: value1' -H 'header2: value1' -H 'header2: value2' -H 'header3:
 value1,value2'
```

El resultado es similar al siguiente:

```
{
  "resource": "/my/path",
  "path": "/my/path",
  "httpMethod": "GET",
  "headers": {
      "header1": "value1",
      "header2": "value2",
      "header3": "value1,value2"
  },
  "multiValueHeaders": {
    "header1": ["value1"],
    "header2": ["value1","value2"],
    "header3": ["value1,value2"]
  },
  "queryStringParameters": {
      "parameter1": "value1",
      "parameter2": "value2",
      "parameter3": "value1,value2"
  },
  "multiValueQueryStringParameters": {
    "parameter1": ["value1"],
    "parameter2": ["value1","value2"],
    "parameter3": ["value1,value2"]
  },
  "requestContext": {
    "accountId": "123456789012",
    "apiId": "id",
    "authorizer": {
```

```
      "claims": null,
      "scopes": null
    },
    "domainName": "id.execute-api.us-east-1.amazonaws.com",
    "domainPrefix": "id",
    "extendedRequestId": "request-id",
    "httpMethod": "GET",
    "identity": {
      "accessKey": null,
      "accountId": null,
      "caller": null,
      "cognitoAuthenticationProvider": null,
      "cognitoAuthenticationType": null,
      "cognitoIdentityId": null,
      "cognitoIdentityPoolId": null,
      "principalOrgId": null,
      "sourceIp": "IP",
      "user": null,
      "userAgent": "user-agent",
      "userArn": null,
      "clientCert": {
        "clientCertPem": "CERT_CONTENT",
        "subjectDN": "www.example.com",
        "issuerDN": "Example issuer",
        "serialNumber": "a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1",
        "validity": {
          "notBefore": "May 28 12:30:02 2019 GMT",
          "notAfter": "Aug  5 09:36:04 2021 GMT"
        }
      }
    },
    "path": "/my/path",
    "protocol": "HTTP/1.1",
    "requestId": "id=",
    "requestTime": "04/Mar/2020:19:15:17 +0000",
    "requestTimeEpoch": 1583349317135,
    "resourceId": null,
    "resourcePath": "/my/path",
    "stage": "$default"
  },
  "pathParameters": null,
  "stageVariables": null,
  "body": "Hello from Lambda!",
  "isBase64Encoded": false
```

```
}
```

> **Note**
>
> En la entrada:
>
> - La clave `headers` solo puede contener encabezados de un solo valor.
> - La clave `multiValueHeaders` puede contener encabezados de varios valores y encabezados de un solo valor.
> - Si especifica valores para `headers` y `multiValueHeaders`, API Gateway los combina en una sola lista. Si se especifica el mismo par de clave-valor en ambos, solo los valores de `multiValueHeaders` aparecerán en la lista combinada.

En la entrada a la función de Lambda de backend, el objeto `requestContext` es un mapa de pares de clave-valor. En cada par, la clave es el nombre de una propiedad de la variable $context y el valor es el valor de esa propiedad. API Gateway podría agregar nuevas claves al mapa.

En función de las características habilitadas, el mapa `requestContext` puede variar de API en API. Por ejemplo, en el ejemplo anterior, no se especifica ningún tipo de autorización, por lo que no hay propiedades `$context.authorizer.*` o `$context.identity.*` presentes. Cuando se especifica un tipo de autorización, esto hace que API Gateway transfiera información del usuario autorizado al punto de conexión de integración en un objeto `requestContext.identity` tal y como se indica a continuación:

- Cuando el tipo de autorización es `AWS_IAM`, la información del usuario autorizado incluye propiedades `$context.identity.*`.
- Cuando el tipo de autorización es `COGNITO_USER_POOLS` (autorizador de Amazon Cognito), la información del usuario autorizado incluye las propiedades `$context.identity.cognito*` y `$context.authorizer.claims.*`.
- Cuando el tipo de autorización es `CUSTOM` (autorizador de Lambda), la información del usuario autorizado incluye las propiedades `$context.authorizer.principalId` y otras propiedades `$context.authorizer.*` aplicables.

Formato de salida de una función de Lambda para la integración de proxy

Con la integración de proxy de Lambda, API Gateway requiere que la función de Lambda del backend devuelva la salida de acuerdo con el siguiente formato JSON:

```
{
    "isBase64Encoded": true|false,
    "statusCode": httpStatusCode,
    "headers": { "headerName": "headerValue", ... },
    "multiValueHeaders": { "headerName": ["headerValue", "headerValue2", ...], ... },
    "body": "..."
}
```

En la salida:

- Las claves `headers` y `multiValueHeaders` pueden no estar especificadas si no se van a devolver más encabezados de respuesta.
- La clave `headers` solo puede contener encabezados de un solo valor.
- La clave `multiValueHeaders` puede contener encabezados de varios valores y encabezados de un solo valor. Puede utilizar la clave `multiValueHeaders` para especificar todos los encabezados adicionales, incluidos los que solo contienen un valor.
- Si especifica valores para `headers` y `multiValueHeaders`, API Gateway los combina en una sola lista. Si se especifica el mismo par de clave-valor en ambos, solo los valores de `multiValueHeaders` aparecerán en la lista combinada.

Para habilitar CORS para la integración de proxy de Lambda, debe agregar `Access-Control-Allow-Origin:`*`domain-name`* a la salida `headers`. *`domain-name`* puede ser `*` para cualquier nombre de dominio. La salida `body` se serializa en el frontend como la carga de respuesta del método. Si `body` es un blob binario, puede codificarlo como una cadena codificada en Base64 estableciendo `isBase64Encoded` en `true` y configurando `*/*` como Binary Media Type (Tipo de medio binario). De lo contrario, puede establecerlo en `false` o dejarlo sin especificar.

> **Note**
>
> Para obtener más información acerca de cómo habilitar la compatibilidad con datos binarios, consulte Habilitar la compatibilidad con datos binarios mediante la consola de API Gateway.

> Para ver una función de Lambda de ejemplo, consulte Devolución de medios binarios desde una integración de proxy de Lambda en API Gateway.

Si la salida de la función tiene un formato diferente, API Gateway devuelve una respuesta de error `502 Bad Gateway`.

Para devolver una respuesta en una función de Lambda de Node.js, puede utilizar comandos como los siguientes:

- Para devolver un resultado correcto, llame a `callback(null, {"statusCode": 200, "body": "results"})`.
- Para producir una excepción, llame a `callback(new Error('internal server error'))`.
- En caso de que se produzca un error del lado del cliente (por ejemplo, si falta un parámetro necesario), puede llamar a `callback(null, {"statusCode": 400, "body": "Missing parameters of ..."})` para devolver el error sin iniciar una excepción.

En una función de Lambda de `async` de Node.js, la sintaxis correspondiente sería:

- Para devolver un resultado correcto, llame a `return {"statusCode": 200, "body": "results"}`.
- Para producir una excepción, llame a `throw new Error("internal server error")`.
- En caso de que se produzca un error del lado del cliente (por ejemplo, si falta un parámetro necesario), puede llamar a `return {"statusCode": 400, "body": "Missing parameters of ..."}` para devolver el error sin iniciar una excepción.

Configuración de una integración de proxy de Lambda para API Gateway mediante la AWS CLI

En esta sección mostramos cómo configurar una API con la integración de proxy de Lambda mediante la AWS CLI. Para obtener instrucciones detalladas acerca de cómo utilizar la consola de API Gateway para configurar un recurso de proxy con la integración de proxy de Lambda, consulte Tutorial: Creación de una API de REST con una integración de proxy de Lambda.

Como ejemplo, utilizaremos la siguiente función de Lambda de muestra como el backend de la API:

```
export const handler = async(event, context) => {
    console.log('Received event:', JSON.stringify(event, null, 2));
```

```
    var res ={
        "statusCode": 200,
        "headers": {
            "Content-Type": "*/*"
        }
    };
    var greeter = 'World';
    if (event.greeter && event.greeter!=="") {
        greeter =  event.greeter;
    } else if (event.body && event.body !== "") {
        var body = JSON.parse(event.body);
        if (body.greeter && body.greeter !== "") {
            greeter = body.greeter;
        }
    } else if (event.queryStringParameters && event.queryStringParameters.greeter &&
 event.queryStringParameters.greeter !== "") {
        greeter = event.queryStringParameters.greeter;
    } else if (event.multiValueHeaders && event.multiValueHeaders.greeter &&
 event.multiValueHeaders.greeter != "") {
        greeter = event.multiValueHeaders.greeter.join(" and ");
    } else if (event.headers && event.headers.greeter && event.headers.greeter != "") {
        greeter = event.headers.greeter;
    }
    res.body = "Hello, " + greeter + "!";
    return res
};
```

Si se compara con la configuración de la integración personalizada de Lambda en the section called “ Configuración de integraciones de Lambda personalizadas ”, la entrada de esta función de Lambda se puede expresar en los parámetros de la solicitud y en el cuerpo. Tiene más libertad para permitir al cliente transferir los mismos datos de entrada. En este caso el cliente puede transferir el nombre de greeter como un parámetro de cadena de consulta, un encabezado o una propiedad del cuerpo. La función también puede admitir la integración de Lambda personalizada. La configuración de la API es más sencilla. No se configura ninguna respuesta de método ni de integración.

Para configurar una integración de proxy de Lambda mediante la AWS CLI

1. Utilice el siguiente comando create-rest-api para crear una API privada:

   ```
   aws apigateway create-rest-api --name 'HelloWorld (AWS CLI)'
   ```

   El resultado será similar al siguiente:

```
{
    "name": "HelloWorldProxy (AWS CLI)",
    "id": "te6si5ach7",
    "rootResourceId" : "krznpq9xpg",
    "createdDate": 1508461860
}
```

En este ejemplo, se utiliza el `id` de API (`te6si5ach7`) y el `rootResourceId` (`krznpq9xpg`).

2. Utilice el siguiente comando create-resource para crear un recurso de `/greeting` de API Gateway:

```
aws apigateway create-resource \
      --rest-api-id te6si5ach7 \
      --parent-id krznpq9xpg \
      --path-part {proxy+}
```

El resultado será similar al siguiente:

```
{
    "path": "/{proxy+}",
    "pathPart": "{proxy+}",
    "id": "2jf6xt",
    "parentId": "krznpq9xpg"
}
```

Debe usar el valor `id` del recurso `{proxy+}` (`2jf6xt`) para crear un método en el recurso `/{proxy+}` en el siguiente paso.

3. Utilice el siguiente comando put-method para crear una solicitud de método `ANY` de `ANY /{proxy+}`:

```
aws apigateway put-method --rest-api-id te6si5ach7 \
       --resource-id 2jf6xt \
       --http-method ANY \
       --authorization-type "NONE"
```

El resultado será similar al siguiente:

```
{
```

```
    "apiKeyRequired": false,
    "httpMethod": "ANY",
    "authorizationType": "NONE"
}
```

Este método de API permite al cliente recibir o enviar saludos de la función de Lambda al backend.

4. Utilice el siguiente comando put-integration para configurar la integración del método `ANY /{proxy+}` con una función de Lambda denominada `HelloWorld`. Esta función responde a la solicitud con un mensaje de `"Hello, {name}!"`, si se proporciona el parámetro `greeter`, o bien `"Hello, World!"`, si no se ha establecido el parámetro de cadena de consulta.

```
aws apigateway put-integration \
      --rest-api-id te6si5ach7 \
      --resource-id 2jf6xt \
      --http-method ANY \
      --type AWS_PROXY \
      --integration-http-method POST \
      --uri arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123456789012:function:HelloWorld/invocations \
      --credentials arn:aws:iam::123456789012:role/apigAwsProxyRole
```

> **Important**
>
> Para las integraciones Lambda, debe utilizar el método HTTP de `POST` para la solicitud de integración, según la especificación de la acción del servicio de Lambda para las invocaciones de funciones. El rol de IAM de `apigAwsProxyRole` debe tener políticas que permitan al servicio `apigateway` invocar funciones Lambda. Para obtener más información sobre los permisos de IAM, consulte the section called “Modelo de permisos de API Gateway para invocar una API”.

El resultado será similar al siguiente:

```
{
    "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"
}
```

En lugar de proporcionar un rol de IAM para `credentials`, puede utilizar el comando add-permission para agregar permisos basados en recursos. Esto es lo que hace la consola de API Gateway.

5. Utilice el siguiente comando create-deployment para implementar la API en una etapa `test`:

```
aws apigateway create-deployment  \
      --rest-api-id te6si5ach7 \
      --stage-name test
```

6. Pruebe la API mediante los siguientes comandos cURL en un terminal.

   Llame a la API con el parámetro de cadena de consulta de `?greeter=jane`:

```
curl -X GET 'https://te6si5ach7.execute-api.us-west-2.amazonaws.com/test/greeting?
greeter=jane'
```

   Llame a la API con un parámetro de encabezado de `greeter:jane`:

```
curl -X GET https://te6si5ach7.execute-api.us-west-2.amazonaws.com/test/hi \
  -H 'content-type: application/json' \
  -H 'greeter: jane'
```

   Llame a la API con un cuerpo de `{"greeter":"jane"}`:

```
curl -X POST https://te6si5ach7.execute-api.us-west-2.amazonaws.com/test/hi \
  -H 'content-type: application/json' \
  -d '{ "greeter": "jane" }'
```

   En todos estos casos, la salida es una respuesta 200 con el siguiente cuerpo de respuesta:

```
Hello, jane!
```

Configuración de un recurso de proxy con la integración de proxy de Lambda con una definición OpenAPI

Para configurar un recurso de proxy con el tipo de integración de proxy de Lambda, cree un recurso de API con un parámetro de ruta expansiva (por ejemplo, `/parent/{proxy+}`) e integre este recurso con un backend de funciones Lambda (por ejemplo, `arn:aws:lambda:us-west-2:123456789012:function:SimpleLambda4ProxyResource`) en el método `ANY`. El parámetro de ruta expansiva debe estar al final de la ruta del recurso de la API. Al igual que con un recurso que no es de proxy, puede configurar el recurso de proxy mediante la consola de API Gateway importando un archivo de definición de OpenAPI o llamando directamente a la API REST de API Gateway.

El siguiente archivo de definición de API de OpenAPI muestra un ejemplo de una API con un recurso de proxy que se integra con la función de Lambda denominada `SimpleLambda4ProxyResource`.

OpenAPI 3.0

```
{
   "openapi": "3.0.0",
   "info": {
      "version": "2016-09-12T17:50:37Z",
      "title": "ProxyIntegrationWithLambda"
   },
   "paths": {
      "/{proxy+}": {
         "x-amazon-apigateway-any-method": {
            "parameters": [
               {
                  "name": "proxy",
                  "in": "path",
                  "required": true,
                  "schema": {
                     "type": "string"
                  }
               }
            ],
            "responses": {},
            "x-amazon-apigateway-integration": {
               "responses": {
                  "default": {
                     "statusCode": "200"
                  }
```

```
               },
               "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-east-1:123456789012:function:SimpleLambda4ProxyResource/
invocations",
               "passthroughBehavior": "when_no_match",
               "httpMethod": "POST",
               "cacheNamespace": "roq9wj",
               "cacheKeyParameters": [
                  "method.request.path.proxy"
               ],
               "type": "aws_proxy"
            }
         }
      }
   },
   "servers": [
      {
         "url": "https://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"
        }
      }
    }
  }
}
```

Con la integración de proxy de Lambda, API Gateway asigna en tiempo de ejecución una solicitud entrante en el parámetro de entrada `event` de la función de Lambda. La entrada incluye el método de solicitud, la ruta, los encabezados, todos los parámetros de cadena de consulta, todas las cargas, el contexto asociado y todas las variables de etapa definidas. El formato de entrada se explica en Formato de entrada de una función de Lambda para la integración de proxy. Para que API Gateway asigne la salida de Lambda a respuestas HTTP correctamente, la función de Lambda debe producir

el resultado en el formato que se describe en Formato de salida de una función de Lambda para la integración de proxy.

Con la integración de proxy de Lambda de un recurso de proxy a través del método ANY, la función de Lambda del backend actúa como el controlador de eventos de todas las solicitudes a través del recurso de proxy. Por ejemplo, para registrar patrones de tráfico, puede hacer que el dispositivo móvil envíe la información de ubicación del país, la ciudad, la calle y el edificio enviando una solicitud con `/state/city/street/house` en la ruta URL del recurso de proxy. La función de Lambda del backend puede analizar la ruta URL e insertar tuplas de ubicación en una tabla de DynamoDB.

Configuración de integraciones de Lambda personalizadas en API Gateway

Para mostrar cómo configurar la integración personalizada de Lambda o sin proxy, creamos una API de API Gateway para exponer el método `GET /greeting?greeter={name}` para invocar una función de Lambda. Utilice uno de los siguientes ejemplos de funciones de Lambda para su API.

Utilice uno de los siguientes ejemplos de funciones de Lambda:

Node.js

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

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

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

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

  // Return a greeting to the caller
  return greeting;
```

```
};
```

Python

```
import json


def lambda_handler(event, context):
    print(event)
    res = {
        "statusCode": 200,
        "headers": {
            "Content-Type": "*/*"
        }
    }

    if event['greeter'] == "":
        res['body'] = "Hello, World"
    elif (event['greeter']):
        res['body'] = "Hello, " + event['greeter'] + "!"
    else:
        raise Exception('Missing the required greeter parameter.')

    return res
```

La función responde con un mensaje de `"Hello, {name}!"` si el valor del parámetro `greeter` es una cadena no vacía. Si el valor `"Hello, World!"` es una cadena vacía, devuelve un mensaje de `greeter`. La función devuelve un mensaje de error de `"Missing the required greeter parameter."` si el parámetro greeter no se ha establecido en la solicitud entrante. Asignamos un nombre a la función `HelloWorld`.

Puede crearla en la consola de Lambda o mediante la AWS CLI. En esta sección, hacemos referencia a esta función mediante los siguientes ARN:

```
arn:aws:lambda:us-east-1:123456789012:function:HelloWorld
```

Una vez establecida la función de Lambda en el backend, proceda a configurar la API.

Para configurar la integración personalizada de Lambda mediante la . AWS CLI

1. Utilice el siguiente comando create-rest-api para crear una API privada:

```
aws apigateway create-rest-api --name 'HelloWorld (AWS CLI)'
```

El resultado será similar al siguiente:

```
{
    "name": "HelloWorld (AWS CLI)",
    "id": "te6si5ach7",
    "rootResourceId" : "krznpq9xpg",
    "createdDate": 1508461860
}
```

En este ejemplo, se utiliza el `id` de API (`te6si5ach7`) y el `rootResourceId` (`krznpq9xpg`).

2. Utilice el siguiente comando create-resource para crear un recurso de `/greeting` de API Gateway:

```
aws apigateway create-resource \
      --rest-api-id te6si5ach7 \
      --parent-id krznpq9xpg \
      --path-part greeting
```

El resultado será similar al siguiente:

```
{
    "path": "/greeting",
    "pathPart": "greeting",
    "id": "2jf6xt",
    "parentId": "krznpq9xpg"
}
```

Debe usar el valor `id` del recurso `greeting` (`2jf6xt`) para crear un método en el recurso `/greeting` en el siguiente paso.

3. Utilice el siguiente comando put-method para crear una solicitud de método de API de `GET /greeting?greeter={name}`:

```
aws apigateway put-method --rest-api-id te6si5ach7 \
       --resource-id 2jf6xt \
       --http-method GET \
       --authorization-type "NONE" \
```

```
        --request-parameters method.request.querystring.greeter=false
```

El resultado será similar al siguiente:

```
{
    "apiKeyRequired": false,
    "httpMethod": "GET",
    "authorizationType": "NONE",
    "requestParameters": {
        "method.request.querystring.greeter": false
    }
}
```

Este método de API permite al cliente recibir saludos de la función de Lambda al backend. El parámetro `greeter` es opcional, ya que el backend debería administrar un intermediario anónimo o bien un intermediario identificado.

4. Utilice el siguiente comando put-method-response para configurar la respuesta `200 OK` a la solicitud de método de `GET /greeting?greeter={name}`:

```
aws apigateway put-method-response \
        --rest-api-id te6si5ach7 \
        --resource-id 2jf6xt \
        --http-method GET \
        --status-code 200
```

5. Utilice el siguiente comando put-integration para configurar la integración del método `GET /greeting?greeter={name}` con una función de Lambda denominada `HelloWorld`. Esta función responde a la solicitud con un mensaje de `"Hello, {name}!"`, si se proporciona el parámetro `greeter`, o bien `"Hello, World!"`, si no se ha establecido el parámetro de cadena de consulta.

```
aws apigateway put-integration \
        --rest-api-id te6si5ach7 \
        --resource-id 2jf6xt \
        --http-method GET \
        --type AWS \
        --integration-http-method POST \
        --uri arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789012:function:HelloWorld/invocations \
```

```
        --request-templates '{"application/json":"{\"greeter\":
\"$input.params('greeter')\"}"}' \
        --credentials arn:aws:iam::123456789012:role/apigAwsProxyRole
```

La plantilla de mapeo proporcionada aquí traslada el parámetro de cadena de consulta `greeter` a la propiedad `greeter` de la carga de JSON. Esto es necesario porque la entrada a una función de Lambda se debe expresar en el cuerpo.

> **Important**
>
> Para las integraciones Lambda, debe utilizar el método HTTP de `POST` para la solicitud de integración, según la especificación de la acción del servicio de Lambda para las invocaciones de funciones. El parámetro `uri` es el ARN de la acción que invoca las funciones.
> El resultado será similar al siguiente:

```
{
    "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"
}
```

El rol de IAM de `apigAwsProxyRole` debe tener políticas que permiten al servicio `apigateway` invocar funciones de Lambda. En lugar de proporcionar un rol de IAM para `credentials`, puede llamar al comando add-permission para agregar permisos basados en recursos. De este modo es como la consola de API Gateway agrega estos permisos.

6. Utilice el siguiente comando put-integration-response para configurar la respuesta de integración y pasar el resultado de la función de Lambda al cliente como respuesta del método `200 OK`:

```
aws apigateway put-integration-response \
```

```
        --rest-api-id te6si5ach7 \
        --resource-id 2jf6xt \
        --http-method GET \
        --status-code 200 \
        --selection-pattern ""
```

Si se establece el patrón de selección en una cadena vacía, la respuesta `200 OK` es la opción predeterminada.

El resultado será similar al siguiente:

```
 {
    "selectionPattern": "",
    "statusCode": "200"
}
```

7. Utilice el siguiente comando create-deployment para implementar la API en una etapa `test`:

```
aws apigateway create-deployment \
        --rest-api-id te6si5ach7 \
        --stage-name test
```

8. Pruebe la API mediante el siguiente comando cURL en un 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'
```

Configurar la invocación asíncrona de la función de Lambda de backend

En la integración de Lambda no de proxy (personalizada), se invoca la función de Lambda de backend, de forma síncrona y de forma predeterminada. Este es el comportamiento deseado para la mayoría de operaciones de API de REST. Algunas aplicaciones, sin embargo, requieren que el trabajo se realice de forma asíncrona (como una operación por lotes o una operación de latencia), normalmente por un componente backend independiente. En este caso, la función de Lambda del backend se invoca de forma asíncrona y el método de la API REST de front-end no devuelve el resultado.

Puede configurar la función de Lambda para una integración de Lambda que no sea proxy para invocarla de forma asíncrona especificando `'Event'` como el tipo de invocación de Lambda. Esto se realiza de la siguiente manera:

Configurar la invocación asíncrona de Lambda en la consola de API Gateway

Para que todas las invocaciones sean asincrónicas:

- En Solicitud de integración, añada un encabezado `X-Amz-Invocation-Type` con un valor estático de `'Event'`.

Para que los clientes decidan si las invocaciones son asincrónicas o sincrónicas:

1. En Solicitud de método, añada un encabezado `InvocationType`.
2. En Solicitud de integración, añada un encabezado `X-Amz-Invocation-Type` con una expresión de mapeo de `method.request.header.InvocationType`.
3. Los clientes pueden incluir el encabezado `InvocationType: Event` en solicitudes de API para invocaciones asincrónicas o `InvocationType: RequestResponse` para invocaciones sincrónicas.

Configurar la invocación asíncrona de Lambda mediante OpenAPI

Para que todas las invocaciones sean asincrónicas:

- Agregue el encabezado `X-Amz-Invocation-Type` a la sección x-amazon-apigateway-integration.

```
"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"
        }
```

Para que los clientes decidan si las invocaciones son asincrónicas o sincrónicas:

1. Agregue el siguiente encabezado en cualquier objeto Path Item de OpenAPI.

```
"parameters" : [ {
"name" : "InvocationType",
"in" : "header",
"schema" : {
  "type" : "string"
}
} ]
```

2. Agregue el encabezado `X-Amz-Invocation-Type` a la sección x-amazon-apigateway-integration.

```
"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. Los clientes pueden incluir el encabezado `InvocationType: Event` en solicitudes de API para invocaciones asincrónicas o `InvocationType: RequestResponse` para invocaciones sincrónicas.

Configuración de la invocación asíncrona de Lambda mediante CloudFormation

Las siguientes plantillas de CloudFormation muestran cómo configurar `AWS::ApiGateway::Method` para las invocaciones asíncronas.

Para que todas las invocaciones sean asincrónicas:

```
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'
```

Para que los clientes decidan si las invocaciones son asincrónicas o sincrónicas:

```
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'
```

Los clientes pueden incluir el encabezado `InvocationType: Event` en solicitudes de API para invocaciones asincrónicas o `InvocationType: RequestResponse` para invocaciones sincrónicas.

Gestionar los errores de Lambda en API Gateway

Para las integraciones de Lambda personalizadas, debe asignar los errores devueltos por Lambda en la respuesta de integración a las respuestas de error de HTTP estándar para sus clientes. De lo contrario, los errores de Lambda se devuelven como respuestas `200 OK` de forma predeterminada y el resultado no es intuitivo para los usuarios de la API.

Hay dos tipos de errores que Lambda puede devolver: errores estándar y errores personalizados. En la API, debe tratarlos de forma diferente.

Con la integración de proxy de Lambda, Lambda debe devolver una salida con el siguiente formato:

```
{
  "isBase64Encoded" : "boolean",
  "statusCode": "number",
  "headers": { ... },
  "body": "JSON string"
}
```

En esta salida, `statusCode` es normalmente `4XX` para un error del cliente y `5XX` para un error del servidor. API Gateway trata estos errores asignando el error de Lambda a una respuesta de error HTTP, de acuerdo con el especificad `statusCode`. Para que API Gateway transmita el tipo de error (por ejemplo, `InvalidParameterException`), como parte de la respuesta al cliente, la función de Lambda debe incluir un encabezado (por ejemplo, `"X-Amzn-ErrorType":"InvalidParameterException"`) en la propiedad `headers`.

Temas

- Gestionar los errores de Lambda estándares en API Gateway.
- Gestionar los errores de Lambda personalizados en API Gateway

Gestionar los errores de Lambda estándares en API Gateway.

Un error estándar de AWS Lambda tiene el siguiente formato:

```
{
  "errorMessage": "<replaceable>string</replaceable>",
  "errorType": "<replaceable>string</replaceable>",
  "stackTrace": [
    "<replaceable>string</replaceable>",
    ...
  ]
}
```

Aquí, `errorMessage` es una expresión de cadena del error. `errorType` es un tipo de error o excepción dependiente del lenguaje. `stackTrace` es una lista de expresiones de cadena que indican la pila de rastreo que conduce a la aparición del error.

Por ejemplo, observe la siguiente función de Lambda en JavaScript (Node.js).

```
export const handler = function(event, context, callback) {
    callback(new Error("Malformed input ..."));
};
```

Esta función devuelve el siguiente error de Lambda estándar, que contiene `Malformed input ...` como el mensaje de error:

```
{
  "errorMessage": "Malformed input ...",
  "errorType": "Error",
  "stackTrace": [
    "export const handler (/var/task/index.js:3:14)"
  ]
}
```

Asimismo, considere la siguiente función de Lambda de Python, que produce una `Exception` con el mismo mensaje de error `Malformed input ...`.

```
def lambda_handler(event, context):
    raise Exception('Malformed input ...')
```

Esta función devuelve el siguiente error de Lambda estándar:

```
{
  "stackTrace": [
    [
      "/var/task/lambda_function.py",
      3,
      "lambda_handler",
      "raise Exception('Malformed input ...')"
    ]
  ],
  "errorType": "Exception",
  "errorMessage": "Malformed input ..."
}
```

Tenga en cuenta que los valores de las propiedades `errorType` y `stackTrace` dependen del lenguaje. El error estándar también se aplica a cualquier objeto de error que sea una extensión del objeto `Error` o una subclase de la clase `Exception`.

Para asignar el error de Lambda estándar a una respuesta del método, primero debe decidir cuál es el código de estado HTTP para un error de Lambda determinado. Seguidamente, debe establecer un patrón de expresiones regulares en la propiedad `selectionPattern` del recurso IntegrationResponse asociado con el código de estado HTTP especificado. En la consola de API Gateway, este `selectionPattern` se denomina Expresión regular de error de Lambda en la sección Respuesta de integración, debajo de cada respuesta de integración.

> **Note**
>
> API Gateway utiliza expresiones regulares de estilo de patrón de Java para el mapeo de respuesta. Para obtener más información, consulte Pattern (Patrón) en la documentación de Oracle.

Por ejemplo, use el siguiente comando put-integration-response para configurar una nueva expresión `selectionPattern`:

```
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
```

Asegúrese de configurar también el código de error correspondiente (`400`) en la respuesta del método. De lo contrario, API Gateway produce una respuesta de error de configuración no válida en tiempo de ejecución.

> **Note**
>
> En el tiempo de ejecución, API Gateway coteja el `errorMessage` de error de Lambda con el patrón de la expresión regular en la propiedad `selectionPattern`. Si hay una coincidencia, API Gateway devuelve el error de Lambda como una respuesta HTTP del código de estado HTTP correspondiente. Si no hay ninguna coincidencia, API Gateway devuelve el error como una respuesta predeterminada o produce una excepción de configuración no válida si no se ha configurado una respuesta predeterminada. Configurar el valor `selectionPattern` a `.*` para una determinada respuesta equivale a restablecer esta respuesta como la predeterminada. Esto se debe a que un patrón de solicitud de este tipo coincidirá con todos los mensajes de error, incluidos los nulos, es decir, cualquier mensaje de error no especificado. El mapeo resultante anula al predeterminado. Si utiliza `.+` como patrón de selección para filtrar respuestas, tenga en cuenta que puede que no coincida con una respuesta que contenga un carácter de nueva línea ('\n).

Para actualizar un valor de `selectionPattern` existente mediante la API REST de la AWS CLI, llame a la operación update-integration-response para reemplazar el valor de ruta de `/selectionPattern` por la expresión regular especificada con el patrón `Malformed*`.

Para establecer la expresión `selectionPattern` con la consola de API Gateway, escriba la expresión en el cuadro de texto Expresión regular de error de Lambda cuando configure o actualice una respuesta de integración de un código de estado HTTP especificado.

Gestionar los errores de Lambda personalizados en API Gateway

En lugar del error estándar descrito en la sección anterior, AWS Lambda le permite devolver un objeto de error personalizado como una cadena JSON. El error puede ser cualquier objeto JSON válido. Por ejemplo, la siguiente función de Lambda en JavaScript (Node.js) devuelve un error personalizado:

```
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));
};
```

Debe activar el objeto `myErrorObj` en una cadena JSON antes de llamar a `callback` para salir de la función. De lo contrario, `myErrorObj` se devuelve como una cadena de `"[object Object]"`. Cuando un método de la API está integrado con la función API Gateway anterior, Lambda recibe una respuesta de integración con la siguiente carga:

```
{
    "errorMessage": "{\"errorType\":\"InternalServerError\",\"httpStatus\":500,
\"requestId\":\"e5849002-39a0-11e7-a419-5bb5807c9fb2\",\"trace\":{\"function\":
\"abc()\",\"line\":123,\"file\":\"abc.js\"}}"
}
```

Al igual que con cualquier respuesta de integración, puede transferir esta respuesta de error tal como está a la respuesta del método. O puede usar una plantilla de mapeo para transformar la carga en un formato diferente. Considere, por ejemplo, la siguiente plantilla de asignación de cuerpo para una respuesta de método del código de estado `500`:

```
{
    errorMessage: $input.path('$.errorMessage');
}
```

Esta plantilla traduce el cuerpo de respuesta de la integración que contiene la cadena JSON del error personalizado en el siguiente cuerpo de respuesta del método. Este cuerpo de respuesta del método contiene el objeto JSON del error personalizado:

```
{
    "errorMessage" : {
        errorType : "InternalServerError",
        httpStatus : 500,
        requestId : context.awsRequestId,
        trace : {
            "function": "abc()",
            "line": 123,
            "file": "abc.js"
        }
    }
};
```

En función de los requisitos de la API, es posible que tenga que pasar algunas o todas las propiedades de errores personalizados como parámetros de encabezado de la respuesta del método. Para ello, puede aplicar asignaciones de errores personalizados desde el cuerpo de respuesta de integración a los encabezados de respuesta del método.

Por ejemplo, la siguiente extensión de OpenAPI define una asignación desde las propiedades `errorMessage.errorType`, `errorMessage.httpStatus`, `errorMessage.trace.function` y `errorMessage.trace` a `error_type`, `error_status`, `error_trace_function` y `error_trace`, respectivamente.

```
"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"
          },
          ...
        }
    }
}
```

En tiempo de ejecución, API Gateway deserializa el parámetro `integration.response.body` cuando realiza mapeos de encabezado. Sin embargo, esta deserialización se aplica únicamente a los mapeos de cuerpo a encabezado para respuestas de errores personalizados de Lambda y no se aplica a los mapeos de cuerpo a cuerpo que utilizan `$input.body`. Con estas asignaciones de cuerpo a encabezado de errores personalizados, el cliente recibe los siguientes encabezados como parte de la respuesta del método, siempre que los encabezados `error_status`, `error_trace`, `error_trace_function` y `error_type` estén declarados en la solicitud del método.

```
"error_status":"500",
"error_trace":"{\"function\":\"abc()\",\"line\":123,\"file\":\"abc.js\"}",
"error_trace_function":"abc()",
"error_type":"InternalServerError"
```

La propiedad `errorMessage.trace` del cuerpo de respuesta de integración es una propiedad compleja. Se asigna al encabezado `error_trace` como una cadena JSON.

Integraciones de HTTP para las API de REST en API Gateway

Puede integrar un método de API con un punto de enlace HTTP mediante la integración de proxy HTTP o la integración HTTP personalizada.

API Gateway admite los siguientes puertos de punto de enlace: 80, 443 y 1024-65535.

Con la integración de proxy, la configuración es sencilla. Solo tiene que establecer el método HTTP y el URI del punto de enlace HTTP, según los requisitos de backend, si no le preocupa la codificación o el almacenamiento en caché del contenido.

Con la integración personalizada, la configuración requiere más pasos. Además de los pasos de configuración de integración de proxy, debe especificar cómo se mapean los datos entrantes de la solicitud a la solicitud de integración y cómo se mapean los datos resultantes de respuesta de la integración a la respuesta del método.

Temas

- Configurar integraciones de proxy HTTP en API Gateway
- Configurar integraciones HTTP personalizadas en API Gateway

Configurar integraciones de proxy HTTP en API Gateway

Para configurar un recurso de proxy con el tipo de integración de proxy HTTP, cree un recurso de API con un parámetro de ruta extensiva (por ejemplo, `/parent/{proxy+}`) e integre este recurso con un punto de enlace HTTP del backend (por ejemplo, `https://petstore-demo-endpoint.execute-api.com/petstore/{proxy}`) en el método `ANY`. El parámetro de ruta expansiva debe estar al final de la ruta del recurso.

Al igual que con un recurso que no es de proxy, puede configurar un recurso de proxy con la integración de proxy HTTP mediante la consola de API Gateway, importando un archivo de definición de OpenAPI o llamando a la API REST de API Gateway directamente. Para obtener instrucciones detalladas acerca de cómo utilizar la consola de API Gateway para configurar un recurso de proxy con la integración HTTP, consulte Tutorial: Creación de una API de REST con integración de proxy HTTP.

El siguiente archivo de definición de API de OpenAPI muestra un ejemplo de una API con un recurso de proxy que está integrado en el sitio web PetStore.

OpenAPI 3.0

```
{
   "openapi": "3.0.0",
   "info": {
      "version": "2016-09-12T23:19:28Z",
      "title": "PetStoreWithProxyResource"
   },
   "paths": {
      "/{proxy+}": {
         "x-amazon-apigateway-any-method": {
            "parameters": [
               {
                  "name": "proxy",
                  "in": "path",
                  "required": true,
                  "schema": {
                     "type": "string"
                  }
               }
            ],
            "responses": {},
            "x-amazon-apigateway-integration": {
               "responses": {
```

```
                  "default": {
                     "statusCode": "200"
                  }
               },
               "requestParameters": {
                  "integration.request.path.proxy": "method.request.path.proxy"
               },
               "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/
{proxy}",
               "passthroughBehavior": "when_no_match",
               "httpMethod": "ANY",
               "cacheNamespace": "rbftud",
               "cacheKeyParameters": [
                  "method.request.path.proxy"
               ],
               "type": "http_proxy"
            }
         }
      }
   },
   "servers": [
      {
         "url": "https://4z9giyi2c1.execute-api.us-east-1.amazonaws.com/{basePath}",
         "variables": {
            "basePath": {
              "default": "/test"
            }
         }
      }
   ]
}
```

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "version": "2016-09-12T23:19:28Z",
    "title": "PetStoreWithProxyResource"
  },
  "host": "4z9giyi2c1.execute-api.us-east-1.amazonaws.com",
  "basePath": "/test",
  "schemes": [
```

```
    "https"
  ],
  "paths": {
    "/{proxy+}": {
      "x-amazon-apigateway-any-method": {
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "proxy",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {},
        "x-amazon-apigateway-integration": {
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "requestParameters": {
            "integration.request.path.proxy": "method.request.path.proxy"
          },
          "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/{proxy}",
          "passthroughBehavior": "when_no_match",
          "httpMethod": "ANY",
          "cacheNamespace": "rbftud",
          "cacheKeyParameters": [
            "method.request.path.proxy"
          ],
          "type": "http_proxy"
        }
      }
    }
  }
}
```

En este ejemplo, una clave de caché se declara en el parámetro de ruta `method.request.path.proxy` del recurso de proxy. Esta es la configuración predeterminada al

crear la API utilizando la consola de API Gateway. La ruta base de la API (`/test`, correspondiente a una etapa) se mapea a la página PetStore del sitio web (`/petstore`). La solicitud de integración replica todo el sitio web de PetStore utilizando la variable de ruta expansiva de la API y el método catch-all `ANY`. Los siguientes ejemplos ilustran esta replicación.

- Establecer **`ANY`** en **`GET`** y **`{proxy+}`** en **`pets`**

  Solicitud de método iniciada desde el frontend:

  ```
  GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets HTTP/1.1
  ```

  Solicitud de integración enviada al backend:

  ```
  GET http://petstore-demo-endpoint.execute-api.com/petstore/pets HTTP/1.1
  ```

  Las instancias en tiempo de ejecución del método `ANY` y el recurso de proxy son válidas. La llamada devolverá una respuesta `200 OK` con la carga que contiene el primer lote de mascotas, tal y como se devuelve desde el backend.

- Establecer **`ANY`** en **`GET`** y **`{proxy+}`** en **`pets?type=dog`**

  ```
  GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets?type=dog
   HTTP/1.1
  ```

  Solicitud de integración enviada al backend:

  ```
  GET http://petstore-demo-endpoint.execute-api.com/petstore/pets?type=dog HTTP/1.1
  ```

  Las instancias en tiempo de ejecución del método `ANY` y el recurso de proxy son válidas. La llamada devolverá una respuesta `200 OK` con la carga que contiene el primer lote de perros especificados, tal y como se devuelve desde el backend.

- Establecer **`ANY`** en **`GET`** y **`{proxy+}`** en **`pets/{petId}`**

  Solicitud de método iniciada desde el frontend:

  ```
  GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets/1 HTTP/1.1
  ```

  Solicitud de integración enviada al backend:

```
GET http://petstore-demo-endpoint.execute-api.com/petstore/pets/1 HTTP/1.1
```

Las instancias en tiempo de ejecución del método ANY y el recurso de proxy son válidas. La llamada devolverá una respuesta `200 OK` con la carga que contiene la mascota especificada, tal y como se devuelve desde el backend.

- Establecer **`ANY`** en **`POST`** y **`{proxy+}`** en **`pets`**

  Solicitud de método iniciada desde el frontend:

```
POST https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets HTTP/1.1
Content-Type: application/json
Content-Length: ...

{
  "type" : "dog",
  "price" : 1001.00
}
```

  Solicitud de integración enviada al backend:

```
POST http://petstore-demo-endpoint.execute-api.com/petstore/pets HTTP/1.1
Content-Type: application/json
Content-Length: ...

{
  "type" : "dog",
  "price" : 1001.00
}
```

  Las instancias en tiempo de ejecución del método ANY y el recurso de proxy son válidas. La llamada devolverá una respuesta `200 OK` con la carga que contiene la mascota recién creada, tal y como se devuelve desde el backend.

- Establecer **`ANY`** en **`GET`** y **`{proxy+}`** en **`pets/cat`**

  Solicitud de método iniciada desde el frontend:

```
GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets/cat
```

Solicitud de integración enviada al backend:

```
GET http://petstore-demo-endpoint.execute-api.com/petstore/pets/cat
```

La instancia en tiempo de ejecución de la ruta del recurso de proxy no se corresponde con un punto de enlace del backend y la solicitud resultante no es válida. Como resultado, se devuelve una respuesta `400 Bad Request` con el siguiente mensaje de error.

```
{
  "errors": [
    {
      "key": "Pet2.type",
      "message": "Missing required field"
    },
    {
      "key": "Pet2.price",
      "message": "Missing required field"
    }
  ]
}
```

- Establecer **`ANY`** en **`GET`** y **`{proxy+}`** en **`null`**

  Solicitud de método iniciada desde el frontend:

  ```
  GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test
  ```

  Solicitud de integración enviada al backend:

  ```
  GET http://petstore-demo-endpoint.execute-api.com/petstore/pets
  ```

  El recurso de destino es el elemento principal del recurso de proxy, pero la instancia en tiempo de ejecución del método `ANY` no está definida en la API de ese recurso. Como resultado, esta solicitud `GET` devuelve una respuesta `403 Forbidden` con el mensaje de error `Missing Authentication Token` devuelto por API Gateway. Si la API expone el método `ANY` o `GET` en el recurso principal (`/`), la llamada devuelve una respuesta `404 Not Found` con el mensaje `Cannot GET /petstore`, tal y como se devuelve desde el backend.

Para cualquier solicitud cliente, si la URL del punto de enlace de destino no es válida o el verbo HTTP es válido pero no es compatible, el backend devuelve una respuesta `404 Not Found`. Para un método HTTP no admitido, se devuelve una respuesta `403 Forbidden`.

Configurar integraciones HTTP personalizadas en API Gateway

Con la integración HTTP personalizada, también conocida como la integración sin proxy, obtiene más control sobre qué datos se transfieren entre un método de API y una integración de API y cómo transferir los datos. Para ello, se utilizan mapeos de datos.

Como parte de la configuración de solicitud de método, debe establecer la propiedad requestParameters en un recurso Method. Esto declara qué parámetros de solicitud de método, que se aprovisionan desde el cliente, se mapean a los parámetros de solicitud de integración o son aplicables a las propiedades de cuerpo antes de que se envíen al backend. A continuación, como parte de la configuración de solicitud de integración, debe establecer la propiedad requestParameters en el recurso correspondiente de Integration para especificar los mapeos entre parámetros. También debe establecer la propiedad requestTemplates para especificar plantillas de mapeo, una para cada tipo de contenido admitido. Dichas plantillas mapean los parámetros de solicitud de método, o el cuerpo, al cuerpo de solicitud de integración.

De forma similar, como parte de la configuración de respuesta de método, debe establecer la propiedad responseParameters en el recurso MethodResponse. Esto declara qué parámetros de respuesta de método, que se van a enviar al cliente, se mapean desde los parámetros de respuesta de integración o ciertas propiedades de cuerpo aplicables que se devolvieron desde el backend. A continuación, configure selectionPattern para elegir una respuesta de integración basada en la respuesta del backend. Para una integración HTTP sin proxy, esto es una expresión regular. Por ejemplo, para asignar todos los códigos de estado HTTP 2xx desde un punto de conexión HTTP proxy a esta asignación de salida, utilice `2\d{2}`.

> **Note**
>
> API Gateway utiliza expresiones regulares de estilo de patrón de Java para el mapeo de respuesta. Para obtener más información, consulte Pattern (Patrón) en la documentación de Oracle.

A continuación, como parte de la configuración de respuesta de integración, debe establecer la propiedad responseParameters en el recurso IntegrationResponse correspondiente para especificar los mapeos entre parámetros. También debe establecer el mapa responseTemplates

para especificar plantillas de mapeo, una para cada tipo de contenido admitido. Dichas plantillas mapean los parámetros de respuesta de integración, o bien las propiedades de cuerpo de respuesta de integración, al cuerpo de respuesta del método.

Para obtener más información sobre la configuración de plantillas de mapeo, consulte Transformaciones de datos para las API de REST en API Gateway.

Transmisión de la respuesta de integración para las integraciones de proxy en API Gateway

Puede configurar la integración de proxy para controlar cómo API Gateway devuelve la respuesta de integración. De forma predeterminada, API Gateway espera a recibir la respuesta completa antes de iniciar la transmisión. Sin embargo, si configura el modo de transferencia de respuestas de la integración en `STREAM`, API Gateway no espera a que se calcule completamente la respuesta para enviarla al cliente. La transmisión de respuestas funciona para todos los tipos de puntos de conexión de la API de REST.

Utilice la transmisión de respuestas para los siguientes casos de uso:

- Reduzca el tiempo para el primer byte (TTFB) para aplicaciones de IA generativa como los chatbots.
- Transmita archivos de imágenes, vídeos o música de gran tamaño sin utilizar una URL prefirmada de S3.
- Realice operaciones de larga duración y, al mismo tiempo, informe sobre el progreso incremental, como los eventos enviados por el servidor (SSE).
- Supere el límite de carga útil de respuesta de 10 MB de API Gateway.
- Supere el límite de 29 segundos de API Gateway sin solicitar un aumento del tiempo de espera de la integración.
- Reciba una carga útil binaria sin configurar los tipos de medios binarios.

Consideraciones sobre la transmisión de la carga útil de respuesta

Es posible que las siguientes consideraciones afecten al uso de la transmisión de la carga útil de respuesta:

- Solo puede utilizar la transmisión de carga útil de respuesta para tipos de integración `HTTP_PROXY` o `AWS_PROXY`. Esto incluye las integraciones de proxy de Lambda y las integraciones privadas que utilizan integraciones de `HTTP_PROXY`.
- La configuración predeterminada del modo de transferencia es `BUFFERED`. Para utilizar la transmisión de respuestas, debe cambiar el modo de transferencia de respuesta a `STREAM`.
- La transmisión de respuestas solo es compatible con las API de REST.
- La transmisión de solicitudes es incompatible.
- Puede transmitir su respuesta durante 15 minutos como máximo.

- Las transmisiones están sujetas a tiempos de espera por inactividad. Para puntos de conexión regionales o privados, el tiempo de espera es de 5 minutos. Para puntos de conexión optimizados para bordes, el tiempo de espera es de 30 segundos.
- Si utiliza la transmisión de respuesta para una API de REST regional con su propia distribución de CloudFront, puede lograr un tiempo de espera por inactividad de más de 30 segundos lo que aumenta el tiempo de espera de respuesta de la distribución de CloudFront. Para obtener más información, consulte Tiempo de espera de respuesta.
- Cuando el modo de transferencia de respuestas está configurado en `STREAM`, API Gateway no admite características que requieran almacenar en búfer toda la respuesta de integración. Por este motivo, las siguientes características no son compatibles con la transmisión de respuestas:
  - Almacenamiento en caché de puntos de conexión
  - Codificación de contenido. Si desea comprimir su respuesta de integración, hágalo en su integración.
  - Transformación de la respuesta con VTL
- Dentro de cada respuesta de transmisión, los primeros 10 MB de carga útil de respuesta no están sujetos a ninguna restricción de ancho de banda. Los datos de carga útil de respuesta que superen los 10 MB están restringidos a 2 MB/s.
- Cuando la conexión entre el cliente y API Gateway o entre API Gateway y Lambda se cierra debido al tiempo de espera, es posible que la función de Lambda continúe ejecutándose. Para obtener más información, consulte Configuración del tiempo de espera de la función de Lambda.
- La transmisión de respuestas tiene un costo. Para obtener más información, consulte Precio de API Gateway.

Configuración de una integración de proxy HTTP con transmisión de respuesta de carga útil en API Gateway

Cuando configura la transmisión de la carga útil de respuesta, especifica el modo de transferencia de respuesta en la solicitud de integración del método. Estos ajustes se configuran en la solicitud de integración para controlar el comportamiento de API Gateway antes y durante la respuesta de integración. Cuando utiliza la transmisión de respuestas, puede configurar el tiempo de espera de la integración en un máximo de 15 minutos.

Cuando utiliza la transmisión de respuestas de carga útil con una integración de `HTTP_PROXY`, API Gateway no enviará el código de estado de respuesta HTTP ni ningún encabezado de respuesta HTTP hasta que reciba todos los encabezados por completo.

Creación de una integración de proxy HTTP con transmisión de respuesta de carga útil

El siguiente procedimiento muestra cómo importar una nueva API con el valor `responseTransferMode` establecido en `STREAM`. Si tiene una API de integración existente y desea modificar `responseTransferMode`, consulte the section called “Actualización del modo de transferencia de respuestas para una integración de proxy HTTP”.

Consola de administración de AWS

Creación de una integración de proxy HTTP con transmisión de respuesta de carga útil

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST.
3. Elija Crear recurso.
4. En Nombre del recurso, escriba **`streaming`**.
5. Elija Crear recurso.
6. Con el recurso /streaming seleccionado, elija Crear método.
7. En Tipo de método, elija CUALQUIERA.
8. Para Tipo de integración, elija HTTP.
9. Elija Integración de proxy HTTP.
10. Para Modo de transferencia de respuesta, elija Flujo.
11. Para Método HTTP, elige un método.
12. Para URL del punto de conexión, ingrese un punto de conexión de integración. Asegúrese de elegir un punto de conexión que genere una gran carga útil para que se la transmita.
13. Elija Crear método.

Después de crear el método, implemente la API.

Para implementar su API

1. Elija Implementar API.
2. En Etapa, seleccione Nueva etapa.
3. En Stage name (Nombre de etapa), escriba **`prod`**.
4. (Opcional) En Description (Descripción), introduzca una descripción.
5. Elija Implementar.

AWS CLI

Creación de una nueva API con transmisión de respuesta de carga útil

1. Copie el siguiente archivo de API abierto y guárdelo como `ResponseStreamDemoSwagger.yaml`. En este archivo, `responseTransferMode` se establece en `STREAM`. El punto de conexión de integración está configurado en `https://example.com`, pero le recomendamos que lo modifique por un punto de conexión que genere una gran carga útil para que se la transmita.

```
openapi: "3.0.1"
info:
  title: "ResponseStreamingDemo"
  version: "2025-04-28T17:28:25Z"
servers:
- url: "{basePath}"
  variables:
    basePath:
      default: "prod"
paths:
  /streaming:
    get:
      x-amazon-apigateway-integration:
        httpMethod: "GET"
        uri: "https://example.com"
        type: "http_proxy"
        timeoutInMillis: 900000
        responseTransferMode: "STREAM"
```

2. Use el comando `import-rest-api` siguiente para importar la definición de OpenAPI:

```
aws apigateway import-rest-api \
  --body 'fileb://~/ResponseStreamDemoSwagger.yaml' \
  --parameters endpointConfigurationTypes=REGIONAL \
  --region us-west-1
```

3. Use el comando `create-deployment` siguiente para implementar la nueva API en una etapa:

```
aws apigateway create-deployment \
  --rest-api-id a1b2c3 \
  --stage-name prod \
```

```
  --region us-west-1
```

Actualización del modo de transferencia de respuestas para una integración de proxy HTTP

El procedimiento siguiente muestra cómo actualizar el modo de transferencia de respuestas para una integración de proxy HTTP.

Consola de administración de AWS

Actualización del modo de transferencia de respuestas para una integración de proxy HTTP

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST.
3. Elija un método.
4. En la pestaña Solicitud de integración, en Configuración de la solicitud de integración, seleccione Editar.
5. Para Modo de transferencia de respuesta, elija Flujo.
6. Seleccione Save.

Después de actualizar el método, implemente la API.

Para implementar su API

1. Elija Implementar API.
2. En Etapa, seleccione Nueva etapa.
3. En Stage name (Nombre de etapa), escriba **`prod`**.
4. (Opcional) En Description (Descripción), introduzca una descripción.
5. Elija Implementar.

AWS CLI

El comando `update-integration` siguiente actualiza el modo de transferencia de una integración de `BUFFERED` a `STREAM`. Para cualquier API existente, el modo de transferencia de respuesta para todas las integraciones se establece en `BUFFERED`.

```
aws apigateway update-integration \
```

```
 --rest-api-id a1b2c3 \
 --resource-id aaa111 \
 --http-method GET \
 --patch-operations "op='replace',path='/responseTransferMode',value=STREAM" \
 --region us-west-1
```

Tiene que volver a implementar la API para que los cambios surtan efecto. Si personalizó el tiempo de espera de la integración, este valor de tiempo de espera se elimina, ya que API Gateway transmite la respuesta durante un máximo de 5 minutos.

El comando `update-integration` siguiente actualiza el modo de transferencia de una integración de `STREAM` a `BUFFERED`:

```
aws apigateway update-integration \
 --rest-api-id a1b2c3 \
 --resource-id aaa111 \
 --http-method GET \
 --patch-operations "op='replace',path='/responseTransferMode',value=BUFFERED" \
 --region us-west-1
```

Tiene que volver a implementar la API para que los cambios surtan efecto.

Configuración de una integración de proxy de Lambda con transmisión de respuesta de carga útil en API Gateway

Puede transmitir la respuesta de una función de Lambda para mejorar el rendimiento del tiempo hasta el primer byte (TTFB) y devolver respuestas parciales al cliente a medida que estén disponibles. API Gateway requiere que utilice la API de Lambda InvokeWithResponseStream para invocar la función de Lambda. API Gateway pasa un objeto de evento a la función de Lambda. La función de Lambda del backend analiza los datos de la solicitud entrante para determinar la respuesta que devuelve. Para que API Gateway transmita la salida de Lambda, la función de Lambda debe producir el formato que requiere API Gateway.

Diferencias en las integraciones de proxy de Lambda entre el modo de transferencia de respuesta en flujo y en búfer

La siguiente lista describe las diferencias entre una integración de proxy de Lambda y una integración de proxy de Lambda para la transmisión de respuestas:

- API Gateway usa la API InvokeWithResponseStream para invocar la integración del proxy de Lambda para la transmisión de respuestas. Esto da como resultado un URI diferente, que es el siguiente:

  ```
  arn:aws:apigateway:us-west-1:lambda:path/2021-11-15/functions/arn:aws:lambda:us-west-1:111122223333:function:my-function-name/response-streaming-invocations
  ```

  Este ARN utiliza una fecha diferente para la versión de la API y una acción de servicio diferente en comparación con la integración del proxy de Lambda.

  Si utiliza la consola de API Gateway para la transmisión de respuestas, la consola utilizará el URI correcto para usted.

- En una integración de proxy de Lambda, API Gateway envía la respuesta al cliente solo después de recibir la respuesta completa de Lambda. En una integración de proxy de Lambda para la transmisión de respuestas, API Gateway inicia la transmisión de carga útil después de recibir los metadatos y el delimitador válidos de Lambda.
- La integración de proxy de Lambda para la transmisión de respuestas utiliza el mismo formato de entrada que la integración de proxy, pero requiere un formato de salida diferente.

Formato de integración de proxy de Lambda para transmisión de respuestas

Cuando API Gateway invoca una función de Lambda con transmisión de respuestas, el formato de entrada es el mismo que el formato de entrada de una función de Lambda para la integración de proxy. Para obtener más información, consulte the section called “Formato de entrada de una función de Lambda para la integración de proxy”.

Cuando Lambda transmite una respuesta a API Gateway, la respuesta debe seguir el siguiente formato. Este formato utiliza un delimitador para separar el JSON de los metadatos y la carga útil sin procesar. En este caso, los datos de carga útil se transmiten tal como los transmite la función de Lambda de transmisión:

```
{
  "headers": {"headerName": "headerValue", ...},
  "multiValueHeaders": { "headerName": ["headerValue", "headerValue2", ...], ... },
  "cookies" : ["cookie1", "cookie2"],
  "statusCode": httpStatusCode
}<DELIMITER>PAYLOAD1 | PAYLOAD2 | PAYLOAD3
```

En la salida:

- Las claves `headers`, `multiValueHeaders`, `cookies` y `statusCode` pueden no estar especificadas si no se van a devolver más encabezados de respuesta.
- La clave `headers` solo puede contener encabezados de un solo valor.
- La salida espera que los encabezados contengan `Transfer-Encoding: chunked` o `Content-length: number`. Si la función no devuelve ninguno de estos encabezados, API Gateway agrega `Transfer-Encoding: chunked` al encabezado de respuesta.
- La clave `multiValueHeaders` puede contener encabezados de varios valores y encabezados de un solo valor. Puede utilizar la clave `multiValueHeaders` para especificar todos los encabezados adicionales, incluidos los que solo contienen un valor.
- Si especifica valores para `headers` y `multiValueHeaders`, API Gateway los combina en una sola lista. Si se especifica el mismo par de clave-valor en ambos, solo los valores de `multiValueHeaders` aparecerán en la lista combinada.
- Los metadatos deben ser un JSON válido. Solo se admiten las claves de `headers`, `multiValueHeaders`, `cookies` y `statusCode`.
- Debe proporcionar un delimitador después del JSON de metadatos. El delimitador debe tener 8 bytes nulos y debe aparecer dentro de los primeros 16 KB de datos de transmisión.
- API Gateway no requiere un formato específico para la carga útil de respuesta de método.

Si utiliza una URL de función para transmitir la función de Lambda, debe modificar la entrada y la salida de la función de Lambda para cumplir estos requisitos.

Si el resultado de la función de Lambda no cumple con los requisitos de este formato, es posible que API Gateway siga invocando la función de Lambda. En la siguiente tabla se muestran las combinaciones de la configuración de las solicitudes de integración de API y el código de función de Lambda que admite API Gateway. Esto incluye las combinaciones compatibles con el modo de transferencia de respuestas en búfer.

Modo de transferencia de respuesta	El código de función se ajusta al formato requerido	API de invocación de Lambda	Compatible con API Gateway
De streaming	Sí	InvokeWithResponseStream	Sí. API Gateway transmite la respuesta .
De streaming	No	InvokeWithResponseStream	No. API Gateway invoca la función de Lambda y devuelve una respuesta de error 500.
De streaming	Sí	Invoke	No. API Gateway no admite esta configuración de integración.
De streaming	No	Invoke	No. API Gateway no admite esta configuración de integración.
Almacenado en búfer	Sí	InvokeWithResponseStream	No. API Gateway no admite esta configuración de integración.
Almacenado en búfer	No	InvokeWithResponseStream	No. API Gateway no admite esta configuración de integración.
Almacenado en búfer	Sí	Invoke	API Gateway devuelve los encabezados HTTP y el código de estado, pero no el cuerpo de la respuesta.

Modo de transferencia de respuesta	El código de función se ajusta al formato requerido	API de invocación de Lambda	Compatible con API Gateway
Almacenado en búfer	No	Invoke	Sí. Esta es una integración de proxy de Lambda. Para obtener más información, consulte Integración de proxy de Lambda.

Configuración de una integración de proxy de Lambda con transmisión de respuesta de carga útil en API Gateway

Cuando configura la transmisión de la carga útil de respuesta, especifica el modo de transferencia en la solicitud de integración del recurso. Estos ajustes se configuran en la solicitud de integración para controlar el comportamiento de API Gateway antes y durante la respuesta de integración.

Ejemplos de funciones de Lambda para la transmisión de respuestas

La función de Lambda debe cumplir el the section called “Formato de integración de proxy de Lambda para transmisión de respuestas”. Le recomendamos que utilice una de las tres funciones de Lambda de ejemplo para probar la transmisión de respuestas. Cuando cree la función de Lambda, asegúrese de hacer lo siguiente:

- Proporcione un tiempo de espera adecuado para la función. Le recomendamos configurar un tiempo de espera de al menos 1 minuto para obtener información sobre la transmisión de respuestas. Al crear los recursos de producción, asegúrese de que el tiempo de espera de la función de Lambda abarque todo el ciclo de solicitud. Para obtener más información, consulte Configuración del tiempo de espera de la función de Lambda.
- Utilice el último tiempo de ejecución de Node.js.
- Utilice una región en la que esté disponible la transmisión de respuestas de Lambda.

Using HttpResponseStream.from

El siguiente ejemplo de código transmite los objetos de metadatos y las cargas útiles de JSON de vuelta al cliente mediante el método `awslambda.HttpResponseStream()` sin utilizar el método de canalización. No es necesario crear el delimitador. Para obtener más información, consulte Escritura de funciones de Lambda habilitadas para la transmisión de respuestas.

```
export const handler = awslambda.streamifyResponse(
  async (event, responseStream, context) => {
    const httpResponseMetadata = {
      "statusCode": 200,
      "headers": {
        "x-foo": "bar"
      },
      "multiValueHeaders": {
        "x-mv1": ["hello", "world"],
        "Set-Cookie": ["c1=blue", "c2=red"]
      }
    };

    responseStream = awslambda.HttpResponseStream.from(responseStream,
 httpResponseMetadata);
    await new Promise(r => setTimeout(r, 1000)); // synthetic delay

    responseStream.write("First payload ");
    await new Promise(r => setTimeout(r, 1000)); // synthetic delay

    responseStream.write("Final payload");
    responseStream.end();
});
```

Using the pipeline method

Lambda recomienda que, al escribir funciones habilitadas para la transmisión de respuestas, utilice el decorador `awslambda.streamifyResponse()` que proporcionan los tiempos de ejecución nativos de Node.js y el método de `pipeline()`. Cuando utiliza el método de canalización, no necesita crear el delimitador; Lambda lo hace automáticamente. Para obtener más información, consulte Escritura de funciones de Lambda habilitadas para la transmisión de respuestas.

El ejemplo de código siguiente transmite los objetos de metadatos y tres cargas útiles de JSON de vuelta al cliente.

```
import { pipeline } from 'node:stream/promises';
import { Readable } from 'node:stream';

export const handler = awslambda.streamifyResponse(
  async (event, responseStream, context) => {
    const httpResponseMetadata = {
      statusCode: 200,
      headers: {
        "Content-Type": "text/plain",
        "X-Custom-Header": "Example-Custom-Header"
      }
    };

    responseStream = awslambda.HttpResponseStream.from(responseStream,
 httpResponseMetadata);

    const dataStream = Readable.from(async function* () {
      yield "FIRST payload\n";
      await new Promise(r => setTimeout(r, 1000));
      yield "SECOND payload\n";
      await new Promise(r => setTimeout(r, 1000));
      yield "THIRD payload\n";
      await new Promise(r => setTimeout(r, 1000));
    }());

    await pipeline(dataStream, responseStream);
  }
);
```

Without using the pipeline method

El ejemplo de código siguiente transmite los objetos de metadatos y tres cargas útiles de JSON de vuelta al cliente sin usar el método `awslambda.HttpResponseStream()`. Sin el método `awslambda.HttpResponseStream()`, debe incluir un delimitador de 8 bytes nulos entre los metadatos y la carga útil.

```
export const handler = awslambda.streamifyResponse(async (event, response, ctx) => {
  response.write('{"statusCode": 200, "headers": {"hdr-x": "val-x"}}');
  response.write("\x00".repeat(8)); // DELIMITER
  await new Promise(r => setTimeout(r, 1000));

  response.write("FIRST payload");
```

```
  await new Promise(r => setTimeout(r, 1000));

  response.write("SECOND payload");
  await new Promise(r => setTimeout(r, 1000));

  response.write("FINAL payload");
  response.end();
});
```

Creación de una integración de proxy de Lambda con transmisión de respuesta de carga útil

En el siguiente procedimiento se muestra cómo crear una integración de proxy de Lambda con la transmisión de respuesta de carga útil. Utilice la función de Lambda de ejemplo o cree la suya propia.

Consola de administración de AWS

Creación de una integración de proxy de Lambda con transmisión de respuesta de carga útil

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST.
3. Elija Crear recurso.
4. En Nombre del recurso, escriba **`streaming`**.
5. Elija Crear recurso.
6. Con el recurso /streaming seleccionado, elija Crear método.
7. En Tipo de método, elija CUALQUIERA.
8. Para Tipo de integración, elija Lambda.
9. Elija Integración de proxy de Lambda.
10. Para Modo de transferencia de respuesta, elija Flujo.
11. Para la función de Lambda, elija el nombre de la función de Lambda.

    La consola de API Gateway utiliza automáticamente la API InvokeWithResponseStream para invocar la función de Lambda. Es responsable de escribir una función de Lambda habilitada para la transmisión de respuestas. Para ver un ejemplo, consulta the section called “Ejemplos de funciones de Lambda para la transmisión de respuestas”.
12. Elija Crear método.

Después de crear el método, implemente la API.

Para implementar su API

1. Elija Implementar API.
2. En Etapa, seleccione Nueva etapa.
3. En Stage name (Nombre de etapa), escriba **`prod`**.
4. (Opcional) En Description (Descripción), introduzca una descripción.
5. Elija Implementar.

AWS CLI

El siguiente procedimiento muestra cómo importar una nueva API con el valor `responseTransferMode` establecido en `STREAM`. Si tiene una API de integración existente y desea modificar `responseTransferMode`, consulte the section called “Actualización del modo de transferencia de respuestas para una integración de proxy de Lambda”.

Creación de una nueva API con transmisión de respuesta de carga útil

1. Copie el siguiente archivo de API abierto y guárdelo como `ResponseStreamDemoSwagger.yaml`. En este archivo, `responseTransferMode` se establece en `STREAM` y el URI de integración se establece en `arn:aws:apigateway:us-west-1:lambda:path/2021-11-15/functions/arn:aws:lambda:us-west-1:111122223333:function:my-function-name/response-streaming-invocations`.

   Sustituya el nombre de la función de `my-function` por una función habilitada para transmisión y sustituya las credenciales por un rol de IAM que tenga políticas que permitan al servicio de `apigateway` invocar funciones de Lambda.

```
openapi: "3.0.1"
info:
  title: "ResponseStreamingDemo"
  version: "2025-04-28T17:28:25Z"
servers:
- url: "{basePath}"
  variables:
    basePath:
      default: "prod"
```

```
paths:
  /lambda:
    get:
      x-amazon-apigateway-integration:
        httpMethod: "POST"
        uri: "arn:aws:apigateway:us-west-1:lambda:path/2021-11-15/functions/
arn:aws:lambda:us-west-1:111122223333:function:my-function-name/response-
streaming-invocations"
        type: "aws_proxy"
        timeoutInMillis: 90000
        responseTransferMode: "STREAM"
        credentials: "arn:aws:iam::111122223333:role/apigateway-lambda-role"
```

En lugar de proporcionar un rol de IAM para credenciales, puede utilizar el comando `add-permission` para Lambda para agregar permisos basados en recursos.

2. Use el comando `import-rest-api` siguiente para importar la definición de OpenAPI:

```
aws apigateway import-rest-api \
  --body 'fileb://~/ResponseStreamDemoSwagger.yaml' \
  --parameters endpointConfigurationTypes=REGIONAL \
  --region us-west-1
```

3. Use el comando `create-deployment` siguiente para implementar la nueva API en una etapa:

```
aws apigateway create-deployment \
  --rest-api-id a1b2c2 \
  --stage-name prod \
  --region us-west-1
```

Actualización del modo de transferencia de respuestas para una integración de proxy de Lambda

El procedimiento siguiente muestra cómo actualizar el modo de transferencia de respuestas para una integración de proxy de Lambda. Cuando cambie el modo de transferencia de respuesta a transmisión, actualice la función de Lambda para que cumpla con los requisitos de la transmisión de respuesta. Utilice la función de Lambda de ejemplo o cree la suya propia.

Consola de administración de AWS

Actualización del modo de transferencia de respuestas para una integración de proxy de Lambda

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST.
3. Elija un método.
4. En la pestaña Solicitud de integración, en Configuración de la solicitud de integración, seleccione Editar.
5. Para Modo de transferencia de respuesta, elija Flujo.
6. Para la función de Lambda, elija el nombre de la función de Lambda.
7. Seleccione Save.

Después de actualizar el método, implemente la API.

Para implementar su API

1. Elija Implementar API.
2. En Etapa, seleccione Nueva etapa.
3. En Stage name (Nombre de etapa), escriba **`prod`**.
4. (Opcional) En Description (Descripción), introduzca una descripción.
5. Elija Implementar.

AWS CLI

1. Actualización de la función de Lambda para que esté habilitada para la transmisión.
2. Utilice el comando AWS CLI siguiente para actualizar el URI de integración y el modo de transferencia de respuestas de la integración:

```
aws apigateway update-integration \
 --rest-api-id a1b2c3 \
 --resource-id aaa111 \
 --http-method ANY \
 --patch-operations "[{\"op\":\"replace\",\"path\":\"/uri\",\"value
\":\"arn:aws:apigateway:us-west-1:lambda:path/2021-11-15/functions/
arn:aws:lambda:us-west-1:111122223333:function:my-function-name/response-
```

```
streaming-invocations\"}, {\"op\":\"replace\",\"path\":\"/responseTransferMode
\",\"value\":\"STREAM\"}]" \
 --region us-west-1
```

3. Vuelva a implementar la API para que los cambios se apliquen.

Solución de problemas relacionados con la transmisión de respuestas en API Gateway

La siguiente guía de solución de problemas puede ayudar a resolver los problemas relacionados con las API que usan transmisión de respuestas.

Solución de problemas generales

Puede usar el TestInvokeMethod o la pestaña de prueba de la consola para probar la respuesta de la transmisión. Es posible que las siguientes consideraciones afecten al uso de la invocación de prueba para la transmisión de respuesta:

- Cuando prueba el método, API Gateway almacena en búfer la carga útil de respuesta transmitida. Una vez que se cumple alguna de las siguientes condiciones, API Gateway devuelve una respuesta única que contiene la carga útil almacenada en búfer:
  - La solicitud está completa
  - Han transcurrido 35 segundos
  - Se ha almacenado en búfer más de 1 MB de carga útil de respuesta
- Si transcurren más de 35 segundos antes de que el método devuelva un estado de respuesta HTTP y todos los encabezados, el estado de respuesta devuelto en TestInvokeMethod es 0.
- API Gateway no produce ningún registro de ejecución.

Una vez implementada la API, puede probar la respuesta de la transmisión mediante un comando curl. Le recomendamos que utilice la opción `-i` para incluir los encabezados de respuesta de protocolo en la salida. Visualización de los datos de respuesta según llegan, uso de la opción de curl `--no-buffer`

Errores de cURL de solución de problemas

Si está probando una integración y recibe el error `curl: (18) transfer closed with outstanding read data remaining`, asegúrese de que el tiempo de espera de la integración sea lo suficientemente largo. Si utiliza una función de Lambda, debe actualizar el tiempo de espera

de respuesta de la función de Lambda. Para obtener más información, consulte Configuración del tiempo de espera de la función de Lambda.

Solución de problemas mediante el registro de acceso

Puede utilizar los registros de acceso de la etapa de API de REST para registrar y solucionar problemas de la transmisión de respuestas. Además de las variables existentes, puede utilizar las variables del registro de acceso siguientes:

`$context.integration.responseTransferMode`

El modo de transferencia de respuesta de la integración. Puede ser `BUFFERED` o `STREAMED`.

`$context.integration.timeToAllHeaders`

El tiempo entre que API Gateway establece la conexión de integración y se reciben del cliente todos los encabezados de respuesta de integración.

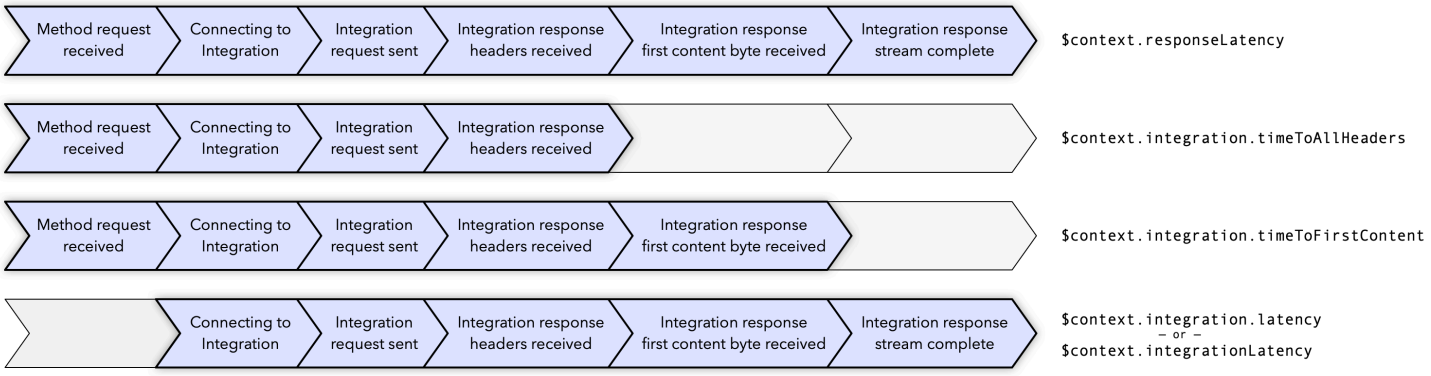
`$context.integration.timeToFirstContent`

El tiempo transcurrido entre el momento en que API Gateway establece la conexión de integración y el momento en que recibe los primeros bytes de contenido.

`$context.integration.latency` o `$context.integrationLatency`

El tiempo entre que API Gateway establece la conexión de integración y se completa la transmisión de respuesta de integración.

La siguiente figura muestra cómo estas variables del registro de acceso representan los diferentes componentes de un flujo de respuesta.



Para obtener más información acerca de los registros de acceso, consulte the section called "CloudWatch Logs". También puede usar X-Ray para supervisar su flujo de respuesta. Para obtener más información, consulte the section called "X-Ray".

Integraciones privadas para las API de REST en API Gateway

Use una integración privada para exponer los recursos HTTP/HTTPS dentro de una Amazon VPC para que puedan acceder los clientes que están fuera de la VPC. Esto amplía el acceso a los recursos de la VPC privada más allá de los límites de esta. Puede controlar el acceso a la API mediante cualquiera de los métodos de autorización admitidos por API Gateway.

Para crear una integración privada, primero debe crear un enlace de la VPC. API Gateway admite enlaces de VPC V2 para las API de REST. Los enlaces de VPC V2 le permiten crear integraciones privadas que conectan la API de REST a los equilibradores de carga de aplicación sin usar un equilibrador de carga de red. El uso de un equilibrador de carga de aplicación le permite conectarse a las aplicaciones basadas en contenedores de Amazon ECS y a muchos otros backend. Los enlaces de VPC V1 se consideran un tipo de integración heredado. Aunque API Gateway los admite, le recomendamos que no cree nuevos enlaces de VPC V1.

Consideraciones

Las siguientes consideraciones pueden afectar al uso de las integraciones privadas:

- Todos los recursos deben pertenecer a la misma Cuenta de AWS. Esto incluye el equilibrador de carga, el enlace de la VPC y la API de REST.
- De forma predeterminada, el tráfico de integración privada utiliza el protocolo HTTP. Para usar HTTPS, especifique un `uri` que contenga un nombre de servidor seguro, como `https://example.com:443/test`.
- En una integración privada, API Gateway incluye la parte de etapa del punto de conexión de la API en la solicitud a los recursos de backend. Por ejemplo, si solicita la etapa `test` de una API, API Gateway incluye `test/path` en la solicitud a la integración privada. Para eliminar el nombre de etapa de la solicitud a los recursos de backend, utilice Asignación de parámetros para crear una invalidación para la variable `$context.requestOverride.path`.
- No se admiten las integraciones privadas con AWS Cloud Map.

Temas

- Configuración de enlaces de VPC V2 en API Gateway
- Configuración de una integración privada

- Integración privada mediante enlaces de VPC V1 (heredado)

Configuración de enlaces de VPC V2 en API Gateway

Los enlaces de la VPC le permiten crear integraciones privadas que conectan las rutas de la API a los recursos privados de una VPC, como equilibradores de carga de aplicaciones o aplicaciones basadas en contenedores de Amazon ECS. Una integración privada utiliza un enlace de VPC V2 para encapsular las conexiones entre API Gateway y los recursos de la VPC de destino. Puede reutilizar enlaces de VPC a través de diferentes recursos y API.

Cuando crea un enlace de la VPC, API Gateway crea y administra interfaces de red elástica para el enlace de VPC V2 en la cuenta. Este proceso puede tardar unos minutos. Cuando un enlace de VPC V2 está listo para usar, su estado cambia de `PENDING` a `AVAILABLE`.

> **Note**
>
> Si no se envía tráfico a través del enlace de la VPC durante 60 días, se convierte en `INACTIVE`. Cuando un enlace de la VPC tiene el estado `INACTIVE`, API Gateway elimina todas las interfaces de red del enlace de la VPC. Esto hace que las solicitudes de API que dependen del enlace de la VPC fallen. Si las solicitudes de la API se reanudan, API Gateway reaprovisiona las interfaces de red. Puede tardar unos minutos en crear las interfaces de red y reactivar el enlace de la VPC. Puede utilizar el estado del enlace de la VPC para supervisar el estado del enlace de la VPC.

Creación de un enlace de VPC V2 mediante la AWS CLI

Para crear un enlace de VPC V2, todos los recursos involucrados deben ser propiedad de la misma cuenta de AWS. El siguiente comando create-vpc-link permite crear un enlace de la VPC:

```
aws apigatewayv2 create-vpc-link --name MyVpcLink \
    --subnet-ids subnet-aaaa subnet-bbbb \
    --security-group-ids sg1234 sg5678
```

> **Note**
>
> Los enlaces de VPC V2 son inmutables. Después de crear un enlace de VPC V2, no puede cambiar sus subredes o grupos de seguridad.

Eliminación de un enlace de VPC V2 mediante la AWS CLI

El siguiente comando delete-vpc-link permite eliminar un enlace de la VPC.

```
aws apigatewayv2 delete-vpc-link --vpc-link-id abcd123
```

Disponibilidad por región

Los enlaces de VPC V2 se admiten en las siguientes regiones y zonas de disponibilidad:

Nombre de la región	Región	Zonas de disponibilidad admitidas
Este de EE. UU. (Ohio)	us-east-2	use2-az1, use2-az2, use2-az3
Este de EE. UU. (Norte de Virginia)	us-east-1	use1-az1, use1-az2, use1-az4, use1-az5, use1-az6
Oeste de EE. UU. (Norte de California)	us-west-1	usw1-az1, usw1-az3
Oeste de EE. UU. (Oregón)	us-west-2	usw2-az1, usw2-az2, usw2-az3, usw2-az4
Asia-Pacífico (Hong Kong)	ap-east-1	ape1-az2, ape1-az3
Asia-Pacífico (Bombay)	ap-south-1	aps1-az1, aps1-az2, aps1-az3
Asia-Pacífico (Osaka)	ap-northeast-3	apne3-az1, apne3-az2, apne3-az3

Nombre de la región	Región	Zonas de disponibilidad admitidas
Asia-Pacífico (Seúl)	ap-northeast-2	apne2-az1, apne2-az2, apne2-az3
Asia-Pacífico (Singapur)	ap-southeast-1	apse1-az1, apse1-az2, apse1-az3
Asia-Pacífico (Sídney)	ap-southeast-2	apse2-az1, apse2-az2, apse2-az3
Asia-Pacífico (Tokio)	ap-northeast-1	apne1-az1, apne1-az2, apne1-az4
Canadá (centro)	ca-central-1	cac1-az1, cac1-az2
Europa (Fráncfort)	eu-central-1	euc1-az1, euc1-az2, euc1-az3
Europa (Irlanda)	eu-west-1	euw1-az1, euw1-az2, euw1-az3
Europa (Londres)	eu-west-2	euw2-az1, euw2-az2, euw2-az3
Europa (París)	eu-west-3	euw3-az1, euw3-az3
Europa (España)	eu-south-2	eus2-az1, eus2-az2, eus2-az3
Europa (Estocolmo)	eu-north-1	eun1-az1, eun1-az2, eun1-az3
Medio Oriente (Baréin)	me-south-1	mes1-az1, mes1-az2, mes1-az3

Nombre de la región	Región	Zonas de disponibilidad admitidas
América del Sur (São Paulo)	sa-east-1	sae1-az1, sae1-az2, sae1-az3
AWS GovCloud (Oeste de EE. UU.)	us-gov-west-1	usgw1-az1, usgw1-az2, usgw1-az3

Configuración de una integración privada

Para crear una integración privada con un equilibrador de carga de aplicación o un equilibrador de carga de red, cree una integración de proxy HTTP, especifique el enlace de VPC V2 que se va a utilizar y proporcione el ARN de un equilibrador de carga de red o un equilibrador de carga de aplicación. De forma predeterminada, el tráfico de integración privada utiliza el protocolo HTTP. Para usar HTTPS, especifique un `uri` que contenga un nombre de servidor seguro, como `https://example.com:443/test`. Para ver un tutorial completo sobre cómo crear una API de REST con una integración privada, consulte the section called “Tutorial: Creación de una API de REST con una integración privada”.

Creación de una integración privada

El procedimiento siguiente muestra cómo crear una integración privada que se conecte a un equilibrador de carga mediante un enlace de VPC V2.

Consola de administración de AWS

Para ver un tutorial sobre cómo crear una integración privada, consulte the section called “Tutorial: Creación de una API de REST con una integración privada”.

AWS CLI

El siguiente comando put-integration permite crear una integración privada que se conecta a un equilibrador de carga mediante un enlace de VPC V2:

```
aws apigateway put-integration \
    --rest-api-id abcdef123 \
```

```
    --resource-id aaa000 \
    --integration-target 'arn:aws:elasticloadbalancing:us-
east-2:111122223333:loadbalancer/app/myLoadBalancerName/1234567891011' \
    --uri 'https://example.com:443/path' \
    --http-method GET \
    --type HTTP_PROXY \
    --integration-http-method GET \
    --connection-type VPC_LINK \
    --connection-id bbb111
```

En lugar de proporcionar directamente el ID de conexión, puede utilizar una variable de etapa. Cuando implementa la API en una etapa, se establece el ID de enlace de VPC V2. El siguiente comando put-integration crea una integración privada mediante una variable de etapa para el ID de enlace de VPC V2:

```
aws apigateway put-integration \
    --rest-api-id abcdef123 \
    --resource-id aaa000 \
    --integration-target 'arn:aws:elasticloadbalancing:us-
east-2:111122223333:loadbalancer/app/myLoadBalancerName/1234567891011' \
    --uri 'https://example.com:443/path' \
    --http-method GET \
    --type HTTP_PROXY \
    --integration-http-method GET \
    --connection-type VPC_LINK \
    --connection-id "\${stageVariables.vpcLinkV2Id}"
```

No olvide utilizar comillas dobles en la expresión de la variable de etapa (${stageVariables.vpcLinkV2Id}) y utilizar caracteres de escape con $.

OpenAPI

Puede configurar una API con la integración privada importando el archivo de OpenAPI de la API. La configuración es similar a las definiciones de OpenAPI de una API con integraciones HTTP, con las siguientes excepciones:

- Debe establecer `connectionType` explícitamente en `VPC_LINK`.
- Debe establecer `connectionId` explícitamente en el ID de un enlace `VpcLinkV2` o en una variable de etapa que haga referencia al ID de un enlace `VpcLinkV2`.
- El parámetro `uri` de la integración privada apunta a un punto de conexión HTTP/HTTPS de la VPC, pero se utiliza para configurar el encabezado `Host` de la solicitud de integración.

- El parámetro `uri` de la integración privada que tiene un punto de conexión HTTPS en la VPC se utiliza para contrastar el nombre de dominio especificado con el del certificado instalado en el punto de conexión de VPC.

Puede utilizar una variable de etapa para hacer referencia al ID de `VpcLinkV2`. También puede asignar el valor del ID directamente a `connectionId`.

El archivo de OpenAPI con formato JSON contiene un ejemplo de una API con un enlace VPC al que hace referencia una variable de etapa (`${stageVariables.vpcLinkIdV2}`):

```
{
  "swagger": "2.0",
  "info": {
    "version": "2017-11-17T04:40:23Z",
    "title": "MyApiWithVpcLinkV2"
  },
  "host": "abcdef123.execute-api.us-west-2.amazonaws.com",
  "basePath": "/test",
  "schemes": [
    "https"
  ],
  "paths": {
    "/": {
      "get": {
        "produces": [
          "application/json"
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "uri": "https://example.com:443/path",
```

```
          "passthroughBehavior": "when_no_match",
          "connectionType": "VPC_LINK",
          "connectionId": "${stageVariables.vpcLinkV2Id}",
          "integration-target": "arn:aws:elasticloadbalancing:us-
east-2:111122223333:loadbalancer/app/myLoadBalancerName/1234567891011",
          "httpMethod": "GET",
          "type": "http_proxy"
        }
      }
    }
  },
  "definitions": {
    "Empty": {
      "type": "object",
      "title": "Empty Schema"
    }
  }
}
```

Actualización de una integración privada

En el siguiente ejemplo se actualiza el enlace de VPC V2 para una integración privada.

Consola de administración de AWS

Actualización de una integración privada

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST con una integración privada.
3. Elija el recurso y el método que utiliza una integración privada.
4. En la pestaña Solicitud de integración, en Configuración de solicitud de integración, elija Editar.
5. Puede editar la configuración de la integración privada. Si actualmente utiliza un enlace de VPC V1, puede cambiar el enlace de la VPC por un enlace de VPC V2.
6. Seleccione Save.
7. Vuelva a implementar la API para que los cambios se apliquen.

AWS CLI

El siguiente comando update-integration actualiza una integración privada para usar un enlace de VPC V2:

```
aws apigateway update-integration \
    --rest-api-id a1b2c3d4e5 \
    --resource-id a1b2c3 \
    --http-method GET \
    --patch-operations "[{\"op\":\"replace\",\"path\":\"/connectionId\",\"value
\":\"pk0000\"}, {\"op\":\"replace\",\"path\":\"/uri\",\"value\":\"http://
example.com\"}, {\"op\":\"replace\",\"path\":\"/integrationTarget\",\"value
\":\"arn:aws:elasticloadbalancing:us-east-2:111122223333:loadbalancer/app/
myLoadBalancerName/1234567891011\"}]"
```

Integración privada mediante enlaces de VPC V1 (heredado)

> **Note**
>
> La siguiente implementación de integraciones privadas utiliza enlaces de VPC V1. Los enlaces de VPC V1 son recursos heredados. Se recomienda utilizar los enlaces de VPC V2 para las API de REST.

Para crear una integración privada, primero debe crear un Network Load Balancer. El Network Load Balancer debe tener un agente de escucha que envíe las solicitudes a los recursos de la VPC. Para mejorar la disponibilidad de la API, asegúrese de que el Network Load Balancer dirija el tráfico a los recursos de varias zonas de disponibilidad de Región de AWS. A continuación, debe crear un vínculo de VPC que utilizará para conectar la API y el Network Load Balancer. Después de crear un vínculo de VPC, se crean integraciones privadas para enviar el tráfico de la API a los recursos de esta a través de dicho vínculo y el Network Load Balancer. El equilibrador de carga de red y la API deben pertenecer a la misma Cuenta de AWS.

Temas

- Configuración de un equilibrador de carga de red para integraciones privadas de API Gateway (heredado)
- Concesión de permisos para API Gateway para crear un enlace de VPC (heredado)

- Configuración de una API de API Gateway con integraciones privadas a través de la AWS CLI (heredado)
- Cuentas de API Gateway utilizadas para integraciones privadas (heredado)

Configuración de un equilibrador de carga de red para integraciones privadas de API Gateway (heredado)

> **Note**
>
> La siguiente implementación de integraciones privadas utiliza enlaces de VPC V1. Los enlaces de VPC V1 son recursos heredados. Se recomienda utilizar los enlaces de VPC V2 para las API de REST.

En el siguiente procedimiento, se describen los pasos necesarios para configurar un Network Load Balancer para integraciones privadas de API Gateway utilizando la consola de Amazon EC2 y se incluyen referencias a instrucciones detalladas sobre cada paso.

Para cada VPC en la que disponga de recursos, solo necesita configurar un NLB y un VPCLink. El NLB admite varios agentes de escucha y grupos de destino por NLB. Puede configurar cada servicio como un agente de escucha específico en el NLB y utilizar un único VPCLink para conectarse al NLB. Al crear la integración privada en API Gateway, defina cada servicio utilizando el puerto específico asignado a cada servicio. Para obtener más información, consulte the section called "Tutorial: Creación de una API de REST con una integración privada". El equilibrador de carga de red y la API deben pertenecer a la misma Cuenta de AWS.

Si desea crear un Network Load Balancer privada para una integración privada con la consola de API Gateway

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon EC2 en https://console.aws.amazon.com/ec2/.
2. Instale un servidor web en una instancia de Amazon EC2. Para ver una configuración de ejemplo, consulte Instalación de un servidor web LAMP en Amazon Linux 2.
3. Cree un Network Load Balancer, registre la instancia EC2 con un grupo de destino y agregue el grupo de destino a un agente de escucha del Network Load Balancer. Para obtener información más detallada, siga las instrucciones que se describen en Introducción a los balanceadores de carga de red.

4. Una vez que se haya creado el equilibrador de carga de red, haga lo siguiente:

   a. Anote el ARN del equilibrador de carga de red. Lo necesitará para crear un enlace VPC en API Gateway e integrar la API con los recurso de la VPC detrás del Network Load Balancer.

   b. Desactive la evaluación del grupo de seguridad para PrivateLink.

      - Para desactivar la evaluación de grupos de seguridad para el tráfico de PrivateLink mediante la consola, puede seleccionar la pestaña Seguridad y, a continuación, Editar. En Configuración de seguridad, seleccione Aplicar reglas de entrada al tráfico de PrivateLink.
      - Utilice el siguiente comando set-security-groups para desactivar la evaluación de grupos de seguridad para el tráfico de PrivateLink:

```
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**
>
> No agregue ninguna dependencia a los CIDR de API Gateway, ya que es probable que cambien sin previo aviso.

Concesión de permisos para API Gateway para crear un enlace de VPC (heredado)

> **Note**
>
> La siguiente implementación de integraciones privadas utiliza enlaces de VPC V1. Los enlaces de VPC V1 son recursos heredados. Se recomienda utilizar los enlaces de VPC V2 para las API de REST.

Para que usted o un usuario de la cuenta puedan crear y mantener un enlace VPC, deben tener permiso para crear, eliminar y consultar configuraciones de servicios de punto de conexión de VPC y examinar los balanceadores de carga. Para conceder este tipo de permisos, siga estos pasos.

Para conceder permisos para crear, actualizar y eliminar un enlace VPC

1. Cree una política de IAM similar a la siguiente:

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "apigateway:POST",
                "apigateway:GET",
                "apigateway:PATCH",
                "apigateway:DELETE"
            ],
            "Resource": [
                "arn:aws:apigateway:us-east-1::/vpclinks",
                "arn:aws:apigateway:us-east-1::/vpclinks/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:DescribeLoadBalancers"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVpcEndpointServiceConfiguration",
                "ec2:DeleteVpcEndpointServiceConfigurations",
                "ec2:DescribeVpcEndpointServiceConfigurations",
                "ec2:ModifyVpcEndpointServicePermissions"
            ],
            "Resource": "*"
        }
    ]
}
```

Si desea habilitar el etiquetado para el enlace de VPC, asegúrese de permitir las operaciones de etiquetado. Para obtener más información, consulte the section called “Permitir operaciones de etiquetado”.

2. Cree o seleccione un rol de IAM y asocie la política anterior al rol.
3. Asigne el rol de IAM a sí mismo o al usuario de la cuenta que está creando los enlaces VPC.

Configuración de una API de API Gateway con integraciones privadas a través de la AWS CLI (heredado)

> **Note**
>
> La siguiente implementación de integraciones privadas utiliza enlaces de VPC V1. Los enlaces de VPC V1 son recursos heredados. Se recomienda utilizar los enlaces de VPC V2 para las API de REST.

En el siguiente tutorial se muestra cómo utilizar la AWS CLI para crear un enlace de VPC y una integración privada. Los siguientes requisitos previos son necesarios:

- Es necesario haber creado y configurado un Equilibrador de carga de red con el origen de la VPC como el objetivo. Para obtener más información, consulte Configuración de un equilibrador de carga de red para integraciones privadas de API Gateway (heredado). Este debe estar en la misma Cuenta de AWS que la API. Necesita el ARN del Equilibrador de carga de red para crear el enlace de la VPC.
- Para crear y administrar un `VpcLink`, necesita los permisos necesarios para crear un `VpcLink` en la API. No necesita los permisos para utilizar el `VpcLink`. Para obtener más información, consulte Concesión de permisos para API Gateway para crear un enlace de VPC (heredado).

Para configurar una API con integración privada a través de la AWS CLI

1. Utilice el siguiente comando create-vpc-link para crear un `VpcLink` que apunte al equilibrador de carga de red especificado:

```
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
```

El resultado de este comando confirma la recepción de la solicitud y muestra el estado `PENDING` para `VpcLink` que se está creando.

```
{
    "status": "PENDING",
    "targetArns": [
        "arn:aws:elasticloadbalancing:us-east-2:123456789012:loadbalancer/net/my-
vpclink-test-nlb/1234567890abcdef"
    ],
    "id": "gim7c3",
    "name": "my-test-vpc-link"
}
```

API Gateway tarda de 2 a 4 minutos en completar la creación de `VpcLink`. Si la operación finaliza correctamente, el valor de `status` es `AVAILABLE`. Puede verificarlo llamando al siguiente comando get-vpc-link:

```
aws apigateway get-vpc-link --vpc-link-id gim7c3
```

Si se produce un error en la operación, obtendrá el estado `FAILED` y `statusMessage` contendrá el mensaje de error. Por ejemplo, si intenta crear un enlace `VpcLink` con un equilibrador de carga de red que ya está asociado a un punto de conexión de VPC, obtendrá lo siguiente en la propiedad `statusMessage`:

```
"NLB is already associated with another VPC Endpoint Service"
```

Después de que se cree `VpcLink` correctamente, puede crear una API e integrarla con el recurso de la VPC a través de `VpcLink`.

Anote el valor de `id` del `VpcLink` recién creado. En este resultado de ejemplo, es `gim7c3`. Lo necesitará para configurar la integración privada.

2. Utilice el siguiente comando create-rest-api para crear un recurso `RestApi` de API Gateway:

```
aws apigateway create-rest-api --name 'My VPC Link Test'
```

Anote el valor de `id` de `RestApi` y el valor de `rootResourceId` de `RestApi` del resultado devuelto. Necesitará este valor para realizar otras operaciones en la API.

A continuación, cree una API con solo un método `GET` en el recurso raíz (`/`) e integre el método con el `VpcLink`.

3. Utilice el siguiente comando put-method para crear el método `GET /`:

```
aws apigateway put-method \
       --rest-api-id  abcdef123 \
       --resource-id skpp60rab7 \
       --http-method GET \
       --authorization-type "NONE"
```

Si no utiliza la integración de proxy con `VpcLink`, también debe configurar al menos un método de respuesta del código de estado `200`. En este caso, utilizará la integración de proxy.

4. Tras crear el método `GET /`, se configura la integración. En el caso de una integración privada, se utiliza el parámetro `connection-id` para proporcionar el ID de `VpcLink`. Puede utilizar una variable de fase o ingresar directamente el ID de `VpcLink`. El parámetro `uri` no se utiliza para direccionar las solicitudes al punto de conexión, sino para configurar el encabezado `Host` y validar el certificado.

   Use the VPC link ID

   Utilice el siguiente comando put-integration para usar el ID de `VpcLink` directamente en la integración:

```
aws apigateway put-integration \
    --rest-api-id abcdef123 \
    --resource-id skpp60rab7 \
    --uri 'http://my-vpclink-test-nlb-1234567890abcdef.us-east-2.amazonaws.com'
 \
    --http-method GET \
    --type HTTP_PROXY \
    --integration-http-method GET \
    --connection-type VPC_LINK \
    --connection-id gim7c3
```

Use a stage variable

Utilice el siguiente comando put-integration para usar una variable de etapa para hacer referencia al ID de enlace de VPC. Cuando implementa la API en una etapa, se establece el ID de enlace de la VPC.

```
aws apigateway put-integration \
    --rest-api-id abcdef123 \
    --resource-id skpp60rab7 \
    --uri 'http://my-vpclink-test-nlb-1234567890abcdef.us-east-2.amazonaws.com'
 \
    --http-method GET \
    --type HTTP_PROXY \
    --integration-http-method GET \
    --connection-type VPC_LINK \
    --connection-id "\${stageVariables.vpcLinkId}"
```

No olvide utilizar comillas dobles en la expresión de la variable de etapa (`${stageVariables.vpcLinkId}`) y utilizar caracteres de escape con `$`.

En cualquier momento, también puede actualizar la integración para cambiar el `connection-id`. Utilice el siguiente comando update-integration para actualizar su integración:

```
 aws apigateway update-integration \
   --rest-api-id abcdef123 \
   --resource-id skpp60rab7 \
   --http-method GET \
   --patch-operations '[{"op":"replace","path":"/
connectionId","value":"${stageVariables.vpcLinkId}"}]'
```

No olvide utilizar una lista JSON con forma de cadena como valor del parámetro `patch-operations`.

Puesto que ha utilizado la integración de proxy privada, ahora la API está lista para la implementación y ejecuciones de pruebas.

5. Si usó la variable de etapa para definir el `connection-id`, necesita implementar la API para probarla. Utilice el siguiente comando create-deployment para implementar la API con una variable de etapa:

```
aws apigateway create-deployment \
    --rest-api-id abcdef123 \
    --stage-name test \
    --variables vpcLinkId=gim7c3
```

Para actualizar la variable de etapa con un ID de `VpcLink` diferente (por ejemplo, *asf9d7*), utilice el siguiente comando update-stage:

```
aws apigateway update-stage \
    --rest-api-id abcdef123 \
    --stage-name test \
    --patch-operations op=replace,path='/variables/vpcLinkId',value='asf9d7'
```

Si codifica la propiedad `connection-id` con el literal del ID de `VpcLink`, no necesita implementar la API para probarla. Use el comando test-invoke-method para probar la API antes de implementarla.

6. Use los siguientes comandos para invocar la API:

```
curl -X GET https://abcdef123.execute-api.us-east-2.amazonaws.com/test
```

Si lo desea, también puede ingresar la URL de invocación de la API en un navegador web para ver los resultados.

Cuentas de API Gateway utilizadas para integraciones privadas (heredado)

Los siguientes ID de cuenta de API Gateway específicos de la región se agregan automáticamente al servicio de punto de conexión de VPC como `AllowedPrincipals` cuando se crea un `VpcLink`.

Región	ID de cuenta
us-east-1	392220576650
us-east-2	718770453195
us-west-1	968246515281
us-west-2	109351309407

Región	ID de cuenta
ca-central-1	796887884028
eu-west-1	631144002099
eu-west-2	544388816663
eu-west-3	061510835048
eu-central-1	474240146802
eu-central-2	166639821150
eu-north-1	394634713161
eu-south-1	753362059629
eu-south-2	359345898052
ap-northeast-1	969236854626
ap-northeast-2	020402002396
ap-northeast-3	360671645888
ap-southeast-1	195145609632
ap-southeast-2	798376113853
ap-southeast-3	652364314486
ap-southeast-4	849137399833
ap-south-1	507069717855
ap-south-2	644042651268
ap-east-1	174803364771
sa-east-1	287228555773

Región	ID de cuenta
me-south-1	855739686837
me-central-1	614065512851

Simulación de integraciones para las API de REST en API Gateway

Amazon API Gateway admite integraciones simuladas para métodos API. Esta característica permite a los desarrolladores de API generar respuestas de API directamente desde API Gateway sin necesidad de un backend de integración. Como desarrollador de una API, puede utilizar esta característica para que los equipos dependientes puedan trabajar con una API antes de que se complete el desarrollo del proyecto. También puede usar esta característica para aprovisionar una página de inicio para la API, en la que se proporcione una descripción general de la API y cómo navegar por ella. Para ver un ejemplo de una página de inicio como esta, consulte la solicitud y la respuesta de integración del método GET en el recurso raíz de la API de ejemplo que se describe en Tutorial: Crear una API de REST importando un ejemplo.

Como desarrollador de una API, puede decidir cómo API Gateway responde a una solicitud de integración simulada. Para ello, configura la solicitud de integración y la respuesta de integración del método para asociar una respuesta a un código de estado determinado. Para que un método con la integración simulada devuelva una respuesta `200`, configure la plantilla de asignación del cuerpo de la solicitud de integración para que devuelva lo siguiente.

```
{"statusCode": 200}
```

Configure una respuesta de integración `200` para que tenga la siguiente plantilla de asignación de cuerpo; por ejemplo:

```
{
    "statusCode": 200,
    "message": "Go ahead without me."
}
```

De igual modo, para que el método devuelva, por ejemplo, una respuesta de error `500`, configure la plantilla de asignación del cuerpo de la solicitud de integración para que devuelva lo siguiente.

```
{"statusCode": 500}
```

Configure una respuesta de integración `500` con, por ejemplo, la siguiente plantilla de asignación:

```
{
    "statusCode": 500,
    "message": "The invoked method is not supported on the API resource."
}
```

Si lo desea, también puede hacer que un método de la integración simulada devuelva la respuesta de integración predeterminada sin definir la plantilla de asignación de solicitudes de integración. La respuesta de integración predeterminada es la única que tiene la opción HTTP status regex (Expresión regular de estado de HTTP) sin definir. Asegúrese de que se establecen los comportamientos de acceso directo apropiados.

> **Note**
>
> Las integraciones simuladas no se han diseñado para admitir plantillas de respuestas grandes. Si las necesita para su caso de uso, considere el uso de una integración de Lambda en su lugar.

Si utiliza una plantilla de asignación de solicitudes de integración, puede insertar la lógica de la aplicación para decidir qué respuesta de integración simulada debe devolverse en función de unas determinadas condiciones. Por ejemplo, puede utilizar un parámetro de consulta `scope` en la solicitud de entrada para determinar si se va a devolver una respuesta de éxito o error:

```
{
  #if( $input.params('scope') == "internal" )
    "statusCode": 200
  #else
    "statusCode": 500
  #end
}
```

De esta forma, el método de la integración simulada permite realizar llamadas internas mientras que otros tipos de llamadas se rechazan con respuestas de error.

En esta sección, se describe cómo utilizar la consola de API Gateway para habilitar la integración simulada para un método de la API.

Temas

- Habilitar la integración simulada a través de la consola de API Gateway

Habilitar la integración simulada a través de la consola de API Gateway

Debe tener un método disponible en API Gateway. Siga las instrucciones en Tutorial: Creación de una API de REST con integración no de proxy HTTP.

1. Elija un recurso de API y seleccione Crear método.

   Para configurar el método, haga lo siguiente:

   a. En Tipo de método, seleccione un método.
   b. En Tipo de integración, seleccione Simulación.
   c. Elija Crear método.
   d. En la pestaña Solicitud de método, en Configuración de solicitud de método, elija Editar.
   e. Elija Parámetros de cadenas de consulta de URL. Elija Añadir cadena de consulta y, en Nombre, introduzca **`scope`**. Este parámetro de consulta determina si el intermediario es o no interno.
   f. Seleccione Save.
2. En la pestaña Respuesta de método, elija Crear respuesta y, a continuación, haga lo siguiente:

   a. En Estado HTTP, escriba **`500`**.
   b. Seleccione Save.
3. En la pestaña Solicitud de integración, en Configuración de la solicitud de integración, seleccione Editar.
4. Elija Plantillas de mapeo y, a continuación, haga lo siguiente:

   a. Elija Add mapping template (Añadir plantilla de asignación).
   b. En Tipo de contenido, ingrese **`application/json`**.
   c. En Cuerpo de la plantilla, introduzca lo siguiente:

```
{
  #if( $input.params('scope') == "internal" )
    "statusCode": 200
  #else
    "statusCode": 500
```

```
  #end
}
```

d. Seleccione Save.

5. En la pestaña Respuesta de integración, en Predeterminado: respuesta, elija Editar.
6. Elija Plantillas de mapeo y, a continuación, haga lo siguiente:

   a. En Tipo de contenido, ingrese **`application/json`**.

   b. En Cuerpo de la plantilla, introduzca lo siguiente:

```
{
    "statusCode": 200,
    "message": "Go ahead without me"
}
```

   c. Seleccione Save.

7. Elija Crear respuesta.

   Para crear una respuesta 500, haga lo siguiente:

   a. En Expresión regular de estado de HTTP, escriba **`5\d{2}`**.

   b. En Estado de respuesta del método, seleccione **`500`**.

   c. Seleccione Save.

   d. En 5\d{2} - Respuesta, seleccione Editar.

   e. Elija Plantillas de mapeo y, a continuación, elija Agregar plantilla de mapeo.

   f. En Tipo de contenido, ingrese **`application/json`**.

   g. En Cuerpo de la plantilla, introduzca lo siguiente:

```
{
    "statusCode": 500,
    "message": "The invoked method is not supported on the API resource."
}
```

   h. Seleccione Save.

8. Elija la pestaña Prueba. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña. Para probar la integración simulada, haga lo siguiente:

a. Escriba `scope=internal` en Cadenas de consulta. Seleccione Probar El resultado de la prueba es:

```
Request: /?scope=internal
Status: 200
Latency: 26 ms
Response Body

{
  "statusCode": 200,
  "message": "Go ahead without me"
}

Response Headers

{"Content-Type":"application/json"}
```

b. Escriba `scope=public` en `Query strings` o déjelo en blanco. Seleccione Probar El resultado de la prueba es:

```
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"}
```

También puede devolver los encabezados de una respuesta de integración simulada agregando primero un encabezado a la respuesta del método y configurando después una asignación de

encabezado en la respuesta de integración. De hecho, es así como la consola de API Gateway permite la compatibilidad con CORS, devolviendo los encabezados requeridos por CORS.

Solicitud de la validación de las API de REST en API Gateway

Puede configurar API Gateway para que realice una validación básica de una solicitud de API antes de continuar con la solicitud de integración. Cuando no se supera la validación, API Gateway invalida inmediatamente la solicitud, devuelve una respuesta de error 400 al intermediario y publica los resultados de la validación en logs de CloudWatch Logs. Esto reduce las llamadas innecesarias al backend. Y lo que es más importante, le permite centrarse en el trabajo de validación específico de su aplicación. Para validar el cuerpo de una solicitud, debe verificar que los parámetros obligatorios de la solicitud sean válidos y no nulos o especificar un esquema de modelo para una validación de datos más complicada.

Temas

- Información general de la validación básica de solicitudes en API Gateway
- Modelos de datos para las API de REST
- Configuración de la validación básica de solicitudes en API Gateway
- AWS CloudFormationPlantilla de de una API de ejemplo con validación de solicitudes básica

Información general de la validación básica de solicitudes en API Gateway

API Gateway puede realizar la validación básica de las solicitudes, de modo que pueda centrarse en la validación específica de la aplicación en el backend. Para la validación, API Gateway verifica una o ambas de las condiciones siguientes:

- Los parámetros de la solicitud necesarios en el URI, la cadena de consulta y los encabezados de una solicitud de entrada están presentes y no están vacíos. API Gateway solo comprueba la existencia de un parámetro y no verifica el tipo ni el formato.
- La carga útil de solicitud aplicable se adhiere a la solicitud de esquema JSON configurada del método para un tipo de contenido determinado. Si no se encuentra ningún tipo de contenido coincidente, no se realiza la validación de la solicitud. Para utilizar el mismo modelo independientemente del tipo de contenido, establezca el tipo de contenido para el modelo de datos en `$default`.

Para habilitar la validación, debe especificar reglas de validación en un validador de solicitudes, agregar el validador al mapa de validadores de solicitudes de la API y asignar el validador a métodos de la API individuales.

> **Note**
>
> La validación del cuerpo de la solicitud y Comportamiento de solicitud de método para cargas útiles sin plantillas de asignación para las API de REST en API Gateway son dos temas distintos. Cuando una carga de solicitud no tiene ningún esquema de modelo coincidente, tiene la opción de elegir acceder directamente a ella o bloquear la carga original. Para obtener más información, consulte Comportamiento de solicitud de método para cargas útiles sin plantillas de asignación para las API de REST en API Gateway.

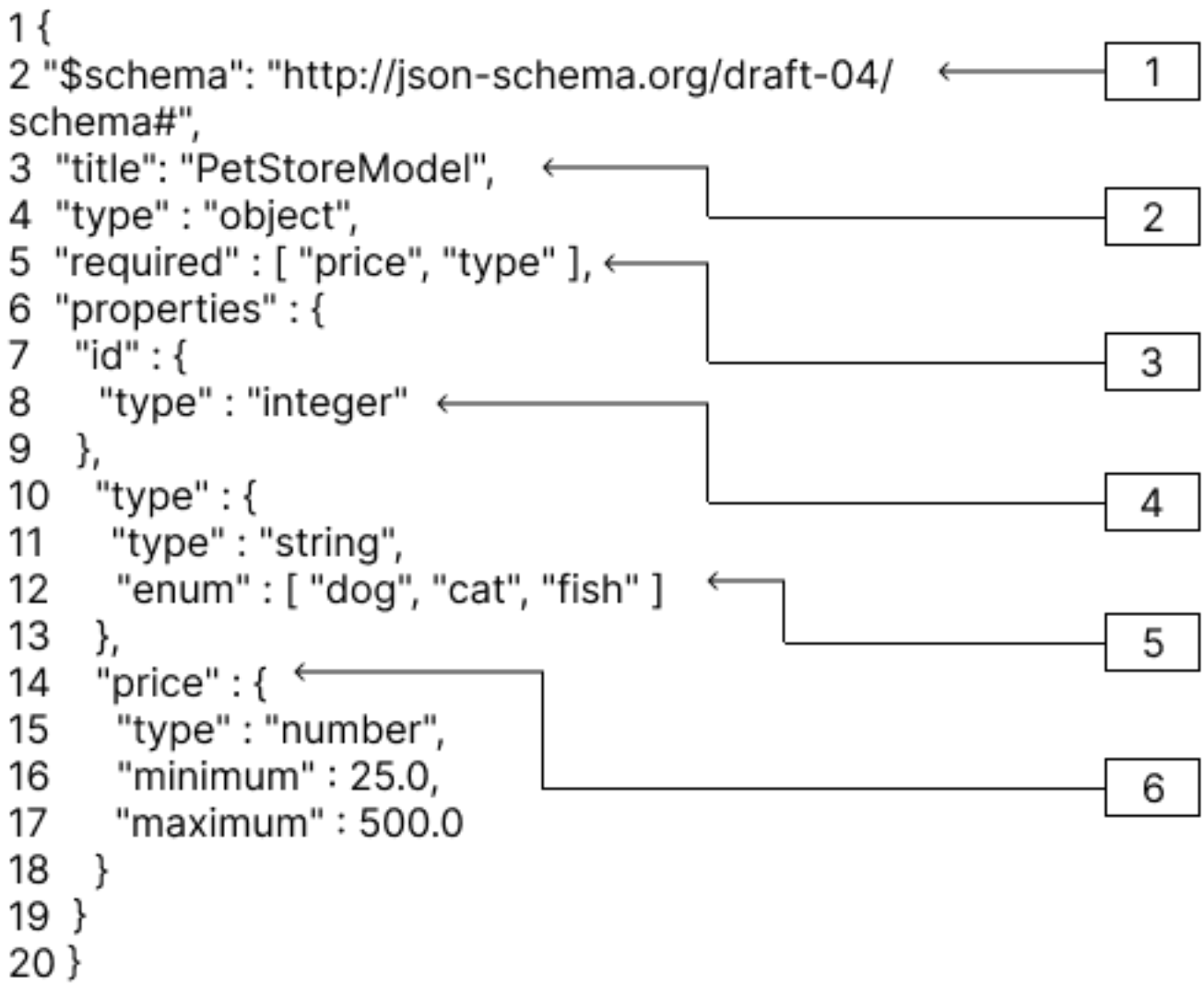
Modelos de datos para las API de REST

En API Gateway, un modelo define la estructura de datos de una carga. En API Gateway, los modelos se definen mediante el borrador 4 del esquema JSON. El siguiente objeto JSON incluye datos de muestra del ejemplo de la tienda de mascotas (Pet Store).

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

Los datos contienen el `id`, `type` y `price` de la mascota. Un modelo de estos datos le permite:

- Utilizar la validación básica de solicitudes.
- Crear plantillas de mapeo para la transformación de datos.
- Crear un tipo de datos definido por el usuario (UDT) al generar un SDK.



En este modelo:

1. El objeto `$schema` representa un identificador de versión válido del esquema JSON. Este esquema es la versión 4 del borrador del esquema JSON.
2. El objeto `title` es un identificador descriptivo del modelo. Este título es `PetStoreModel`.
3. La palabra clave de validación `required` requiere `type` y `price` para la validación básica de solicitudes.
4. Las `properties` del modelo son `id`, `type` y `price`. Cada objeto tiene propiedades que se describen en el modelo.
5. El objeto `type` solo puede tener los valores `dog`, `cat` o `fish`.

6. El objeto `price` es un número y está restringido con un valor `minimum` de 25 y un valor `maximum` de 500.

Modelo PetStore

```
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 }
```

En este modelo:

1. En la línea 2, el objeto `$schema` representa un identificador de versión válido del esquema JSON. Este esquema es la versión 4 del borrador del esquema JSON.
2. En la línea 3, el objeto `title` es un identificador descriptivo del modelo. Este título es `PetStoreModel`.
3. En la línea 5, la palabra clave de validación `required` requiere `type` y `price` para la validación básica de solicitudes.
4. En las líneas 6 a 17, las `properties` del modelo son `id`, `type` y `price`. Cada objeto tiene propiedades que se describen en el modelo.
5. En la línea 12, el objeto `type` solo puede tener los valores `dog`, `cat` o `fish`.
6. En las líneas 14 a 17, el objeto `price` es un número y está restringido con un valor `minimum` de 25 y un valor `maximum` de 500.

Creación de modelos más complejos

Puede utilizar la primitiva `$ref` para crear definiciones reutilizables para modelos más largos. Por ejemplo, puede crear una definición llamada `Price` en la sección `definitions` que describa el objeto `price`. El valor de `$ref` es la definición de `Price`.

```
{
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "PetStoreModelReUsableRef",
  "required" : ["price", "type" ],
  "type" : "object",
  "properties" : {
    "id" : {
      "type" : "integer"
    },
    "type" : {
      "type" : "string",
      "enum" : [ "dog", "cat", "fish" ]
    },
    "price" : {
        "$ref": "#/definitions/Price"
    }
  },
  "definitions" : {
      "Price": {
        "type" : "number",
        "minimum" : 25.0,
        "maximum" : 500.0
            }
      }
}
```

También puede hacer referencia a otro esquema de modelo definido en un archivo de modelo externo. Defina el valor de la propiedad `$ref` en la ubicación del modelo. En el siguiente ejemplo, el modelo `Price` se define en el modelo `PetStorePrice` de la API `a1234`.

```
{
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "PetStorePrice",
  "type": "number",
  "minimum": 25,
  "maximum": 500
```

```
}
```

El modelo más largo puede hacer referencia al modelo `PetStorePrice`.

```
{
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "PetStoreModelReusableRefAPI",
  "required" : [ "price", "type" ],
  "type" : "object",
  "properties" : {
    "id" : {
      "type" : "integer"
    },
    "type" : {
      "type" : "string",
      "enum" : [ "dog", "cat", "fish" ]
    },
    "price" : {
        "$ref": "https://apigateway.amazonaws.com/restapis/a1234/models/PetStorePrice"
    }
  }
}
```

Uso de modelos de datos de salida

Si transforma sus datos, puede definir un modelo de carga en la respuesta de integración. Se puede utilizar un modelo de carga al generar un SDK. Para lenguajes con establecimiento inflexible de tipos, como Java, Objective-C o Swift, el objeto se corresponde con un tipo de datos definido por el usuario (UDT). API Gateway crea un UDT si le facilita un modelo de datos al generar un SDK. Para obtener más información sobre las transformaciones de datos, consulte Asignación de las transformaciones de plantillas de asignación para las API de REST en API Gateway.

En el siguiente ejemplo se muestran datos de salida de una respuesta de integración.

```
{
[
  {
    "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
  }
]
}
```

En el siguiente ejemplo se muestra un modelo de carga que describe los datos de salida.

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

Con este modelo, puede llamar a un SDK para recuperar los valores de las propiedades `description` y `askingPrice` al leer las propiedades `PetStoreOutputModel[i].description` y `PetStoreOutputModel[i].askingPrice`. Si no se proporciona un modelo, API Gateway utiliza el modelo vacío para crear un UDT predeterminado.

Siguientes pasos

- En esta sección se proporcionan recursos que puede utilizar para obtener más información sobre los conceptos presentados en este tema.

  Puede seguir los tutoriales de validación de solicitudes:

  - Configuración de la validación de solicitudes mediante la consola de API Gateway
  - Configuración de la validación básica de solicitudes mediante la AWS CLI

  - Configuración de la validación básica de solicitudes con una definición de OpenAPI
- Para obtener más información sobre la transformación de datos y las plantillas de asignación, Asignación de las transformaciones de plantillas de asignación para las API de REST en API Gateway.

Configuración de la validación básica de solicitudes en API Gateway

En esta sección se muestra cómo configurar la validación de solicitudes para API Gateway mediante la consola, la AWS CLI y una definición de OpenAPI.

Temas

- Configuración de la validación de solicitudes mediante la consola de API Gateway
- Configuración de la validación básica de solicitudes mediante la AWS CLI
- Configuración de la validación básica de solicitudes con una definición de OpenAPI

Configuración de la validación de solicitudes mediante la consola de API Gateway

Para utilizar la consola de API Gateway para validar una solicitud, seleccione uno de los tres validadores para una solicitud de API:

- Validar el cuerpo.
- Validar los parámetros de la cadena de consulta y los encabezados.
- Validar el cuerpo, los parámetros de la cadena de consulta y los encabezados.

Al aplicar uno de los validadores en un método de la API, la consola de API Gateway lo agrega al mapa RequestValidators de la API.

Para seguir este tutorial, utilizará una plantilla CloudFormation para crear una API de API Gateway incompleta. Esta API incompleta tiene un recurso `/validator` con los métodos `GET` y `POST`. Ambos métodos están integrados con el punto de conexión HTTP `http://petstore-demo-endpoint.execute-api.com/petstore/pets`. Configurará dos tipos de validación de solicitudes:

- En el método `GET`, configurará la validación de solicitudes para los parámetros de la cadena de consulta URL.
- En el método `POST`, configurará la validación de solicitudes para el cuerpo de la solicitud.

Esto permitirá que solo ciertas llamadas a la API pasen a la API.

Descargue y descomprima la plantilla de creación de aplicaciones para CloudFormation. Usará esta plantilla para crear una API incompleta. Finalizará el resto de los pasos en la consola de API Gateway.

Para crear una pila de CloudFormation

1. Abra la consola de CloudFormation en https://console.aws.amazon.com/cloudformation.
2. Seleccione Create stack (Crear pila) y, a continuación, seleccione With new resources (standard) (Con nuevos recursos [estándar]).
3. En Specify template (Especificar plantilla), elija Upload a template file (Cargar un archivo de plantilla).
4. Seleccione la plantilla que ha descargado.
5. Elija Next (Siguiente).
6. En Stack name (Nombre de pila), escriba **`request-validation-tutorial-console`** y, a continuación, elija Next (Siguiente).
7. En Configure stack options (Configurar opciones de pila), elija Next (Siguiente).
8. Para Capabilities (Capacidades), sepa que CloudFormation puede crear recursos de IAM en su cuenta.
9. Elija Siguiente y, a continuación, elija Enviar.

CloudFormation aprovisiona los recursos especificados en la plantilla. Puede tardar varios minutos en finalizar el aprovisionamiento de los recursos. Cuando el estado de la pila de CloudFormation sea CREATE_COMPLETE, estará listo para continuar con el paso siguiente.

Para seleccionar la API recién creada

1. Seleccione la pila **`request-validation-tutorial-console`** recién creada.
2. Seleccione Recursos.
3. En ID físico, seleccione la API. Este enlace lo dirigirá a la consola de API Gateway.

Antes de modificar los métodos `GET` y `POST`, debe crear un modelo.

Para crear un modelo

1. Se requiere un modelo para usar la validación de solicitudes en el cuerpo de una solicitud entrante. Para crear un modelo, en el panel de navegación principal, elija Modelos.
2. Seleccione Crear modelo.
3. En Nombre, escriba **`PetStoreModel`**.
4. En Tipo de contenido, indique **`application/json`**. Si no se encuentra ningún tipo de contenido coincidente, no se realiza la validación de la solicitud. Para utilizar el mismo modelo independientemente del tipo de contenido, introduzca **`$default`**.
5. En Descripción, indique **`My PetStore Model`** como descripción del modelo.
6. En Esquema del modelo, pegue el siguiente modelo en el editor de código y seleccione Crear.

```
{
  "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
    }
  }
}
```

Para obtener más información acerca del modelo, consulte Modelos de datos para las API de REST.

Para configurar la validación de solicitudes para un método **GET**

1. En el panel de navegación principal, seleccione Recursos y, a continuación, seleccione el método GET.
2. En la pestaña Solicitud de método, en Configuración de solicitud de método, elija Editar.
3. En Validador de solicitud, seleccione Validar parámetros de cadena de consulta y encabezados.
4. En Parámetros de cadenas de consulta de URL, haga lo siguiente:
   a. Elija Añadir cadena de consulta.
   b. En Nombre, escriba **petType**.
   c. Active la opción Obligatorio.
   d. Mantenga Almacenamiento en caché desactivado.
5. Seleccione Save.
6. En la pestaña Solicitud de integración, en Configuración de la solicitud de integración, seleccione Editar.
7. En Parámetros de cadenas de consulta de URL, haga lo siguiente:
   a. Elija Añadir cadena de consulta.
   b. En Nombre, escriba **petType**.
   c. En Asignado desde, introduzca **method.request.querystring.petType**. Esto mapea el **petType** con el tipo de mascota.

      Para obtener más información sobre el mapeo de datos, consulte el tutorial de mapeo de datos.
   d. Mantenga Almacenamiento en caché desactivado.
8. Seleccione Save.

Para probar la validación de solicitudes para el método **GET**

1. Elija la pestaña Prueba. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña.
2. En Cadenas de consulta, introduzca **petType=dog** y seleccione Prueba.
3. La prueba del método dará como resultado `200 OK` y mostrará una lista de perros.

Para obtener información sobre cómo transformar estos datos de salida, consulte el tutorial de mapeo de datos.

4. Elimine **`petType=dog`** y seleccione Pruebas.
5. La prueba del método devolverá un error `400` con el siguiente mensaje de error:

```
{
  "message": "Missing required request parameters: [petType]"
}
```

Para configurar la validación de solicitudes para el método **`POST`**

1. En el panel de navegación principal, seleccione Recursos y, a continuación, seleccione el método POST.
2. En la pestaña Solicitud de método, en Configuración de solicitud de método, elija Editar.
3. En Validador de solicitudes, seleccione Validar cuerpo.
4. En Cuerpo de la solicitud, seleccione Agregar modelo.
5. En Tipo de contenido, ingrese **`application/json`**. Si no se encuentra ningún tipo de contenido coincidente, no se realiza la validación de la solicitud. Para utilizar el mismo modelo independientemente del tipo de contenido, introduzca `$default`.

   En Modelo, seleccione PetStoreModel.
6. Seleccione Save.

Para probar la validación de solicitudes para un método **`POST`**

1. Elija la pestaña Prueba. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña.
2. En Cuerpo de la solicitud, pegue lo siguiente en el editor de código:

```
{
  "id": 2,
  "name": "Bella",
  "type": "dog",
  "price": 400
}
```

Seleccione Probar

3. La prueba del método devolverá `200 OK` y un mensaje de éxito.
4. En Cuerpo de la solicitud, pegue lo siguiente en el editor de código:

```
{
  "id": 2,
  "name": "Bella",
  "type": "dog",
  "price": 4000
}
```

Seleccione Probar

5. La prueba del método devolverá un error `400` con el siguiente mensaje de error:

```
{
 "message": "Invalid request body"
}
```

En la parte inferior de los registros de pruebas, se mostrará el motivo por el que el cuerpo de la solicitud no es válido. En este caso, el precio de la mascota estaba fuera del máximo especificado en el modelo.

Para eliminar una pila de CloudFormation

1. Abra la consola de CloudFormation en https://console.aws.amazon.com/cloudformation.
2. Seleccione su pila de CloudFormation.
3. Elija Delete (Eliminar) y, a continuación, confirme su elección.

Siguientes pasos

- Para obtener información sobre cómo transformar estos datos de salida y realizar más mapeos de datos, consulte el tutorial de mapeo de datos.
- Siga el tutorial Configuración de la validación básica de solicitudes mediante la AWS CLI para realizar pasos similares con la AWS CLI.

Configuración de la validación básica de solicitudes mediante la AWS CLI

Puede crear un validador para configurar la validación de solicitudes mediante la AWS CLI. Para seguir este tutorial, utilizará una plantilla CloudFormation para crear una API de API Gateway incompleta.

> **Note**
>
> Esta no es la misma plantilla de CloudFormation que la del tutorial de la consola.

Con un recurso `/validator` preexpuesto, creará los métodos `GET` y `POST`. Ambos métodos están integrados con el punto de conexión HTTP `http://petstore-demo-endpoint.execute-api.com/petstore/pets`. Configurará las dos validaciones de solicitudes siguientes:

- En el método `GET`, creará un validador `params-only` para validar los parámetros de la cadena de consulta de URL.
- En el método `POST`, creará un validador `body-only` para validar el cuerpo de la solicitud.

Esto permitirá que solo ciertas llamadas a la API pasen a la API.

Para crear una pila de CloudFormation

Descargue y descomprima la plantilla de creación de aplicaciones para CloudFormation.

Para completar el siguiente tutorial, necesita la versión 2 de la AWS Command Line Interface (AWS CLI).

Para comandos largos, se utiliza un carácter de escape (\) para dividir un comando en varias líneas.

> **Note**
>
> En Windows, algunos comandos de la CLI de Bash que utiliza habitualmente (por ejemplo, `zip`) no son compatibles con los terminales integrados del sistema operativo. Para obtener una versión de Ubuntu y Bash integrada con Windows, instale el subsistema de Windows para Linux. Los comandos de la CLI de ejemplo de esta guía utilizan el formato Linux. Los comandos que incluyen documentos JSON en línea deben reformatearse si utiliza la CLI de Windows.

1. Utilice el siguiente comando para crear la pila CloudFormation.

```
aws cloudformation create-stack --stack-name request-validation-tutorial-cli
 --template-body file://request-validation-tutorial-cli.zip --capabilities
 CAPABILITY_NAMED_IAM
```

2. CloudFormation aprovisiona los recursos especificados en la plantilla. Puede tardar varios minutos en finalizar el aprovisionamiento de los recursos. Use el siguiente comando para ver el estado de su pila CloudFormation.

```
aws cloudformation describe-stacks --stack-name request-validation-tutorial-cli
```

3. Cuando el estado de su pila CloudFormation sea `StackStatus: "CREATE_COMPLETE"`, utilice el siguiente comando para recuperar los valores de salida relevantes para los pasos futuros.

```
aws cloudformation describe-stacks --stack-name request-validation-tutorial-cli
--query "Stacks[*].Outputs[*].{OutputKey: OutputKey, OutputValue: OutputValue,
Description: Description}"
```

   A continuación se muestran los valores de salida:

   - ApiID, que es el identificador de la API. Para este tutorial, el ID de la API es `abc123`.
   - ResourceId, que es el identificador del recurso del validador donde están expuestos los métodos `GET` y `POST`. Para este tutorial, el ID del recurso es `efg456`.

Para crear los validadores de solicitudes e importar un modelo

1. Se requiere un validador para utilizar la validación de solicitudes con la AWS CLI. Utilice el siguiente comando para crear un validador que valide únicamente los parámetros de la solicitud.

```
aws apigateway create-request-validator --rest-api-id abc123 \
      --no-validate-request-body \
      --validate-request-parameters \
      --name params-only
```

   Anote el ID del validador `params-only`.

2. Utilice el siguiente comando para crear un validador que valide únicamente el cuerpo de la solicitud.

```
aws apigateway create-request-validator --rest-api-id abc123 \
      --validate-request-body \
      --no-validate-request-parameters \
      --name body-only
```

Anote el ID del validador `body-only`.

3. Se requiere un modelo para usar la validación de solicitudes en el cuerpo de una solicitud entrante. Utilice el siguiente comando para importar un modelo.

```
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}}}}'
```

Si no se encuentra ningún tipo de contenido coincidente, no se realiza la validación de la solicitud. Para utilizar el mismo modelo independientemente del tipo de contenido, especifique `$default` como la clave.

Para crear los métodos **`GET`** y **`POST`**

1. Use el siguiente comando para añadir el método HTTP `GET` en el recurso `/validate`. Este comando crea el método `GET`, agrega el validador `params-only` y establece la cadena de consulta `petType` según sea necesario.

```
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"
```

Use el siguiente comando para añadir el método HTTP `POST` en el recurso `/validate`. Este comando crea el método `POST`, agrega el validador `body-only` y adjunta el modelo al validador exclusivo para el cuerpo.

```
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. Use el siguiente comando para configurar la respuesta `200 OK` del método `GET /validate`.

```
aws apigateway put-method-response --rest-api-id abc123  \
            --resource-id efg456 \
            --http-method GET \
            --status-code 200
```

Use el siguiente comando para configurar la respuesta `200 OK` del método `POST /validate`.

```
aws apigateway put-method-response --rest-api-id abc123  \
            --resource-id efg456 \
            --http-method POST \
            --status-code 200
```

3. Use el siguiente comando para configurar una `Integration` con un punto de conexión HTTP especificado para el método `GET /validation`.

```
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'
```

Use el siguiente comando para configurar una `Integration` con un punto de conexión HTTP especificado para el método `POST /validation`.

```
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. Use el siguiente comando para configurar la respuesta de integración para el método `GET /validation`.

```
aws apigateway put-integration-response --rest-api-id abc123 \
              --resource-id efg456\
              --http-method GET \
              --status-code 200 \
              --selection-pattern ""
```

   Use el siguiente comando para configurar la respuesta de integración para el método `POST /validation`.

```
aws apigateway put-integration-response --rest-api-id abc123 \
            --resource-id efg456 \
            --http-method POST \
            --status-code 200 \
            --selection-pattern ""
```

Para probar el API

1. Para probar el método `GET`, que realizará la validación de la solicitud para las cadenas de consulta, utilice el siguiente comando:

```
aws apigateway test-invoke-method --rest-api-id abc123 \
            --resource-id efg456 \
            --http-method GET \
            --path-with-query-string '/validate?petType=dog'
```

   El resultado devolverá `200 OK` y una lista de perros.

2. Use el siguiente comando para probar sin incluir la cadena de consulta `petType`.

```
aws apigateway test-invoke-method --rest-api-id abc123 \
            --resource-id efg456 \
            --http-method GET
```

   El resultado devolverá un error `400`.

3. Para probar el método `POST`, que realizará la validación de la solicitud para el cuerpo de la solicitud, utilice el siguiente comando:

```
aws apigateway test-invoke-method --rest-api-id abc123 \
           --resource-id efg456 \
           --http-method POST \
           --body '{"id": 1, "name": "bella", "type": "dog", "price" : 400 }'
```

El resultado devolverá `200 OK` y un mensaje de éxito.

4. Use el siguiente comando para probar el uso de un cuerpo no válido.

```
aws apigateway test-invoke-method --rest-api-id abc123 \
            --resource-id efg456 \
            --http-method POST \
            --body '{"id": 1, "name": "bella", "type": "dog", "price" : 1000 }'
```

El resultado devolverá un error `400`, ya que el precio del perro supera el precio máximo definido por el modelo.

Para eliminar una pila de CloudFormation

- Use el siguiente comando para eliminar los recursos CloudFormation.

```
aws cloudformation delete-stack  --stack-name request-validation-tutorial-cli
```

Configuración de la validación básica de solicitudes con una definición de OpenAPI

Puede declarar un validador de solicitudes en el nivel de la API especificando un conjunto de objetos Objeto x-amazon-apigateway-request-validators.requestValidator en el mapa Objeto x-amazon-apigateway-request-validators para seleccionar qué parte de la solicitud se validará. En la definición de OpenAPI de ejemplo hay dos validadores:

- El validador `all`, que valida tanto el cuerpo, utilizando el modelo de datos `RequestBodyModel`, como los parámetros.

  El modelo de datos `RequestBodyModel` exige que el objeto JSON de entrada contenga las propiedades `name`, `type` y `price`. La propiedad `name` puede ser cualquier cadena, `type` debe ser uno de los campos de la enumeración especificada (`["dog", "cat", "fish"]`) y `price` debe estar comprendido entre 25 y 500. El parámetro `id` no es obligatorio.
- `param-only`, que valida solo los parámetros.

Para habilitar un validador de solicitudes en todos los métodos de una API, especifique una propiedad Propiedad x-amazon-apigateway-request-validator en el nivel de la API de la definición de OpenAPI. En el ejemplo de definición de OpenAPI, el validador `all` se usa en todos los métodos de la API, a menos que se invalide de otro modo. Cuando se utiliza un modelo para validar el cuerpo, si no se encuentra ningún tipo de contenido coincidente, no se realiza la validación de la solicitud. Para utilizar el mismo modelo independientemente del tipo de contenido, especifique `$default` como la clave.

Para habilitar un validador de solicitudes para un método individual, especifique la propiedad `x-amazon-apigateway-request-validator` en el nivel de método. En el ejemplo de definición de OpenAPI, el validador `param-only` anula el validador `all` del método `GET`.

Para importar el ejemplo de OpenAPI a API Gateway, consulte las siguientes instrucciones para Importación de una API regional en API Gateway o para Importación de una API optimizada para bordes en API Gateway.

OpenAPI 3.0

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "ReqValidators Sample",
    "version" : "1.0.0"
  },
  "servers" : [ {
    "url" : "/{basePath}",
    "variables" : {
      "basePath" : {
        "default" : "/v1"
      }
    }
  } ],
  "paths" : {
    "/validation" : {
      "get" : {
        "parameters" : [ {
          "name" : "q1",
          "in" : "query",
          "required" : true,
          "schema" : {
            "type" : "string"
```

```
          }
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response",
            "headers" : {
              "test-method-response-header" : {
                "schema" : {
                  "type" : "string"
                }
              }
            },
            "content" : {
              "application/json" : {
                "schema" : {
                  "$ref" : "#/components/schemas/ArrayOfError"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-request-validator" : "params-only",
        "x-amazon-apigateway-integration" : {
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
          "responses" : {
            "default" : {
              "statusCode" : "400",
              "responseParameters" : {
                "method.response.header.test-method-response-header" : "'static
value'"
              },
              "responseTemplates" : {
                "application/xml" : "xml 400 response template",
                "application/json" : "json 400 response template"
              }
            },
            "2\\d{2}" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.querystring.type" : "method.request.querystring.q1"
          },
```

```
        "passthroughBehavior" : "when_no_match",
        "type" : "http"
      }
    },
    "post" : {
      "parameters" : [ {
        "name" : "h1",
        "in" : "header",
        "required" : true,
        "schema" : {
          "type" : "string"
        }
      } ],
      "requestBody" : {
        "content" : {
          "application/json" : {
            "schema" : {
              "$ref" : "#/components/schemas/RequestBodyModel"
            }
          }
        },
        "required" : true
      },
      "responses" : {
        "200" : {
          "description" : "200 response",
          "headers" : {
            "test-method-response-header" : {
              "schema" : {
                "type" : "string"
              }
            }
          },
          "content" : {
            "application/json" : {
              "schema" : {
                "$ref" : "#/components/schemas/ArrayOfError"
              }
            }
          }
        }
      },
      "x-amazon-apigateway-request-validator" : "all",
      "x-amazon-apigateway-integration" : {
```

```
          "httpMethod" : "POST",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
          "responses" : {
            "default" : {
              "statusCode" : "400",
              "responseParameters" : {
                "method.response.header.test-method-response-header" : "'static
value'"
              },
              "responseTemplates" : {
                "application/xml" : "xml 400 response template",
                "application/json" : "json 400 response template"
              }
            },
            "2\\d{2}" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.header.custom_h1" : "method.request.header.h1"
          },
          "passthroughBehavior" : "when_no_match",
          "type" : "http"
        }
      }
    }
  },
  "components" : {
    "schemas" : {
      "RequestBodyModel" : {
        "required" : [ "name", "price", "type" ],
        "type" : "object",
        "properties" : {
          "id" : {
            "type" : "integer"
          },
          "type" : {
            "type" : "string",
            "enum" : [ "dog", "cat", "fish" ]
          },
          "name" : {
            "type" : "string"
          },
          "price" : {
```

```
            "maximum" : 500.0,
            "minimum" : 25.0,
            "type" : "number"
          }
        }
      },
      "ArrayOfError" : {
        "type" : "array",
        "items" : {
          "$ref" : "#/components/schemas/Error"
        }
      },
      "Error" : {
        "type" : "object"
      }
    }
  },
  "x-amazon-apigateway-request-validators" : {
    "all" : {
      "validateRequestParameters" : true,
      "validateRequestBody" : true
    },
    "params-only" : {
      "validateRequestParameters" : true,
      "validateRequestBody" : false
    }
  }
}
```

OpenAPI 2.0

```
{
  "swagger" : "2.0",
  "info" : {
    "version" : "1.0.0",
    "title" : "ReqValidators Sample"
  },
  "basePath" : "/v1",
  "schemes" : [ "https" ],
  "paths" : {
    "/validation" : {
      "get" : {
        "produces" : [ "application/json", "application/xml" ],
```

```
        "parameters" : [ {
          "name" : "q1",
          "in" : "query",
          "required" : true,
          "type" : "string"
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response",
            "schema" : {
              "$ref" : "#/definitions/ArrayOfError"
            },
            "headers" : {
              "test-method-response-header" : {
                "type" : "string"
              }
            }
          }
        },
        "x-amazon-apigateway-request-validator" : "params-only",
        "x-amazon-apigateway-integration" : {
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
          "responses" : {
            "default" : {
              "statusCode" : "400",
              "responseParameters" : {
                "method.response.header.test-method-response-header" : "'static
value'"
              },
              "responseTemplates" : {
                "application/xml" : "xml 400 response template",
                "application/json" : "json 400 response template"
              }
            },
            "2\\d{2}" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.querystring.type" : "method.request.querystring.q1"
          },
          "passthroughBehavior" : "when_no_match",
          "type" : "http"
```

```
        }
      },
      "post" : {
        "consumes" : [ "application/json" ],
        "produces" : [ "application/json", "application/xml" ],
        "parameters" : [ {
          "name" : "h1",
          "in" : "header",
          "required" : true,
          "type" : "string"
        }, {
          "in" : "body",
          "name" : "RequestBodyModel",
          "required" : true,
          "schema" : {
            "$ref" : "#/definitions/RequestBodyModel"
          }
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response",
            "schema" : {
              "$ref" : "#/definitions/ArrayOfError"
            },
            "headers" : {
              "test-method-response-header" : {
                "type" : "string"
              }
            }
          }
        },
        "x-amazon-apigateway-request-validator" : "all",
        "x-amazon-apigateway-integration" : {
          "httpMethod" : "POST",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
          "responses" : {
            "default" : {
              "statusCode" : "400",
              "responseParameters" : {
                "method.response.header.test-method-response-header" : "'static
value'"
              },
              "responseTemplates" : {
                "application/xml" : "xml 400 response template",
```

```
                "application/json" : "json 400 response template"
              }
            },
            "2\\d{2}" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.header.custom_h1" : "method.request.header.h1"
          },
          "passthroughBehavior" : "when_no_match",
          "type" : "http"
        }
      }
    }
  },
  "definitions" : {
    "RequestBodyModel" : {
      "type" : "object",
      "required" : [ "name", "price", "type" ],
      "properties" : {
        "id" : {
          "type" : "integer"
        },
        "type" : {
          "type" : "string",
          "enum" : [ "dog", "cat", "fish" ]
        },
        "name" : {
          "type" : "string"
        },
        "price" : {
          "type" : "number",
          "minimum" : 25.0,
          "maximum" : 500.0
        }
      }
    },
    "ArrayOfError" : {
      "type" : "array",
      "items" : {
        "$ref" : "#/definitions/Error"
      }
    },
```

```
    "Error" : {
      "type" : "object"
    }
  },
  "x-amazon-apigateway-request-validators" : {
    "all" : {
      "validateRequestParameters" : true,
      "validateRequestBody" : true
    },
    "params-only" : {
      "validateRequestParameters" : true,
      "validateRequestBody" : false
    }
  }
}
```

AWS CloudFormationPlantilla de de una API de ejemplo con validación de solicitudes básica

La siguiente definición de plantilla de ejemplo de CloudFormation define una API de ejemplo con la validación de solicitudes habilitada. La API es un subconjunto de la API PetStore. Expone un método `POST` para añadir una mascota a la colección `pets` y un método `GET` para consultar las mascotas de un tipo especificado.

Hay dos validadores de solicitudes declarados:

`GETValidator`

Este validador está habilitado en el método `GET`. Permite a API Gateway verificar que el parámetro de consulta obligatorio (`q1`) está incluido y no está en blanco en la solicitud de entrada.

`POSTValidator`

Este validador está habilitado en el método `POST`. Permite a API Gateway verificar que el formato de la solicitud de carga se adhiere al `RequestBodyModel` especificado cuando el tipo de contenido es `application/json` si no se encuentra un tipo de contenido coincidente, no se realiza la validación de la solicitud. Para utilizar el mismo modelo independientemente del tipo de contenido, especifique `$default`. `RequestBodyModel` contiene un modelo adicional, `RequestBodyModelId`, para definir el ID de mascota.

```
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:
```

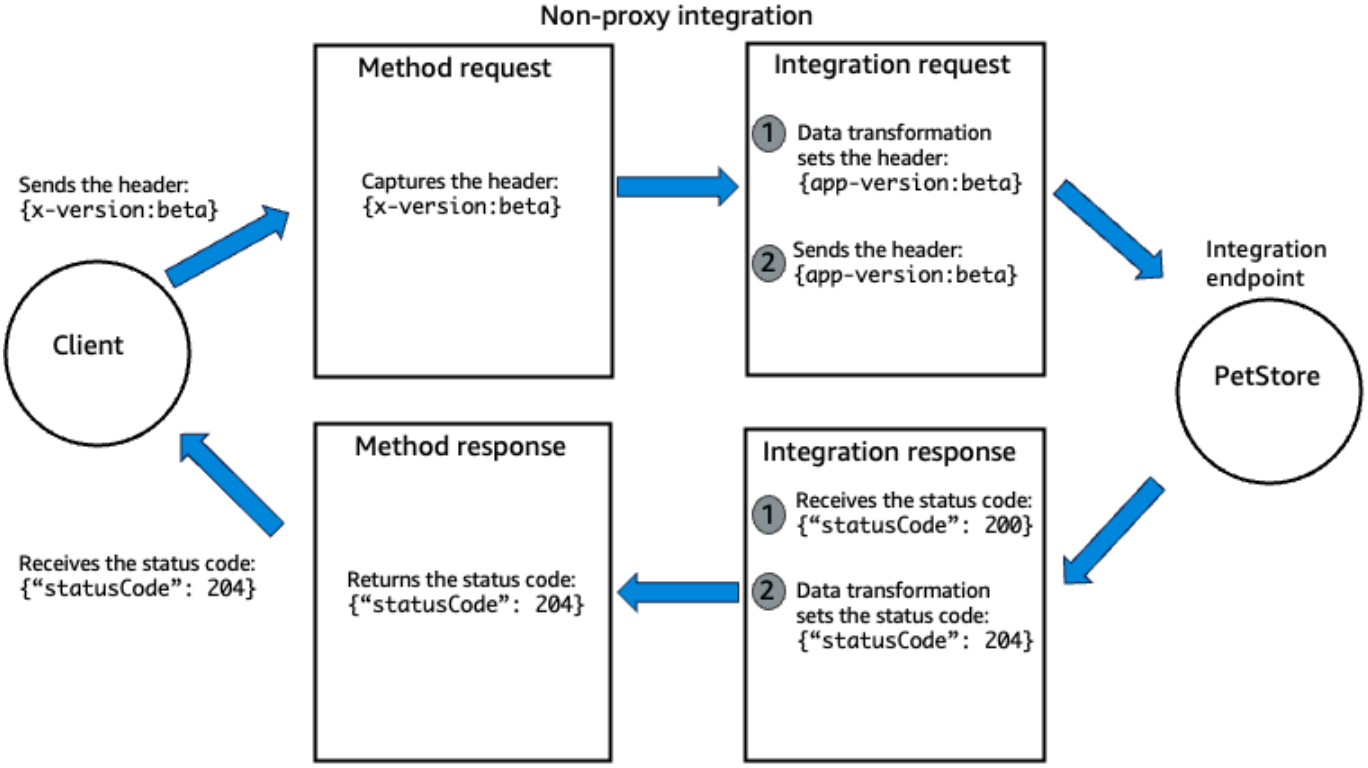
```
        Type: HTTP_PROXY
        IntegrationHttpMethod: GET
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/
  ValidationMethodPost:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref ValidationResource
      HttpMethod: POST
      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}'
```

Transformaciones de datos para las API de REST en API Gateway

> **Note**
>
> En esta sección se explican las características que se utilizan con una integración sin proxy. No obstante, recomendamos que, siempre que sea posible, utilice una integración de proxy para la API de REST. Una integración de proxy HTTP tiene una configuración de integración simplificada y puede evolucionar con el backend sin tener que eliminar la configuración existente. Para obtener más información, consulte the section called “Selección de un tipo de integración de la API”.

Si utiliza una integración sin proxy, puede utilizar dos características de API Gateway para transformar la solicitud de método y la respuesta de integración. Podría transformar la solicitud de método si esta tiene un formato de carga útil diferente al de la carga útil de la solicitud de integración. Podría transformar la respuesta de integración si devuelve un formato de carga útil diferente al formato que necesita devolver en la respuesta del método. Para obtener más información sobre el ciclo de vida de la solicitud, consulte the section called “Recurso de ejemplo para una API de REST”.

En el siguiente ejemplo se muestra una transformación de datos en la que, para el encabezado `"x-version:beta"`, el parámetro de encabezado `x-version` se transforma en el parámetro de encabezado `app-version`. La transformación de datos de `x-version` a `app-version` se produce en la solicitud de integración. De esta manera, el punto de conexión de integración recibe el valor del parámetro de encabezado transformado. Cuando el punto de conexión de integración devuelve un código de estado, este se transforma de `200` a `204` antes de la respuesta de método.



Para crear una transformación de datos, puede utilizar las siguientes características:

Asignación de parámetros

En la asignación de parámetros, puede modificar los parámetros de ruta de URL de solicitud de integración, los parámetros de cadena de consulta de URL o los valores de encabezado HTTP, pero no puede modificar la carga útil de solicitud de integración. También puede modificar

los valores de encabezado de respuesta HTTP. Utilice la asignación de parámetros para crear valores de encabezado estáticos para el uso compartido de recursos entre orígenes (CORS).

Puede usar la asignación de parámetros en la solicitud de integración para integraciones de proxy y de no proxy, pero para usar la asignación de parámetros para una respuesta de integración, necesita una integración de no proxy. La asignación de parámetros no requiere ningún script en Velocity Template Language (VTL). Para obtener más información, consulte the section called "Asignación de parámetros".

Transformaciones de plantillas de asignación

En las transformaciones de plantillas de asignación, se utiliza una plantilla de asignación para asignar parámetros de ruta de URL, parámetros de cadena de consulta de URL, encabezados HTTP y el cuerpo de solicitud de integración o de respuesta de integración. Una plantilla de asignación es un script expresado en Velocity Template Language (VTL) mediante expresiones JSONPath y aplicado a la carga útil en función del encabezado `Content-type`.

Con una plantilla de asignación, puede hacer lo siguiente:

- Seleccionar los datos que desee enviar mediante la integración con Servicios de AWS, como las funciones de Amazon DynamoDB o de Lambda o los puntos de conexión HTTP. Para obtener más información, consulte the section called "Tutorial: modificación de la solicitud y respuesta de integración para integraciones con servicios de AWS".
- Anular condicionalmente los parámetros de solicitud de integración y de respuesta de integración de una API, crear nuevos valores de encabezado y anular códigos de estado. Para obtener más información, consulte the section called "Anulación de los parámetros de solicitud y respuesta y de los códigos de estado de la API".

También puede especificar el comportamiento de la API cuando el cuerpo de una solicitud de integración tenga un encabezado `Content-type` sin plantillas de asignación que coincidan. Esto se denomina comportamiento de acceso directo a la integración. Para obtener más información, consulte the section called "Comportamiento de solicitud de método para cargas útiles sin plantillas de asignación".

Elección entre la asignación de parámetros y las transformaciones de plantillas de asignación

Le recomendamos que utilice la asignación de parámetros para transformar los datos siempre que sea posible. Si la API requiere que cambie el cuerpo o que realice modificaciones y anulaciones

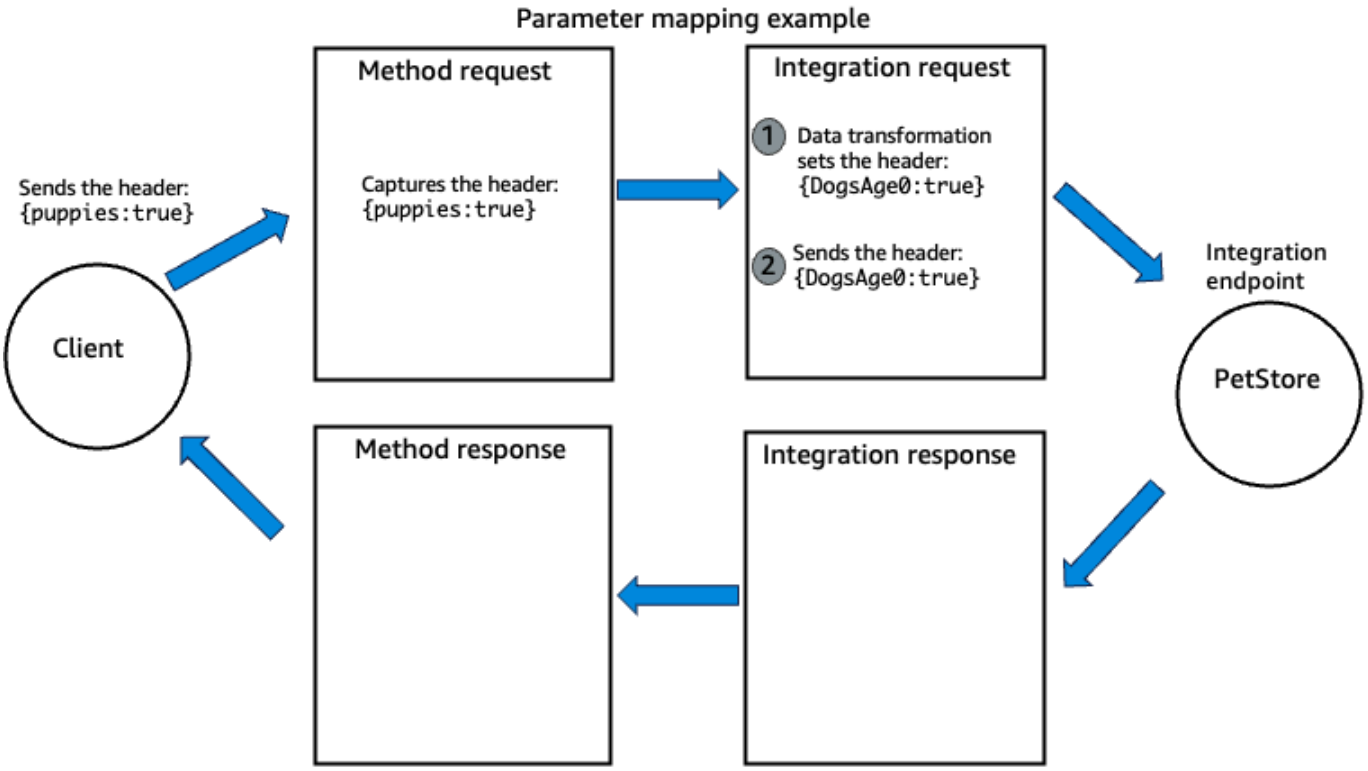
condicionales basadas en la solicitud de integración entrante o en la respuesta de integración y no puede utilizar una integración de proxy, utilice las transformaciones de plantillas de asignación.

Asignación de parámetros para las API de REST en API Gateway

> **Note**
>
> Si utiliza una API de HTTP, consulte the section called “Asignación de parámetros”.

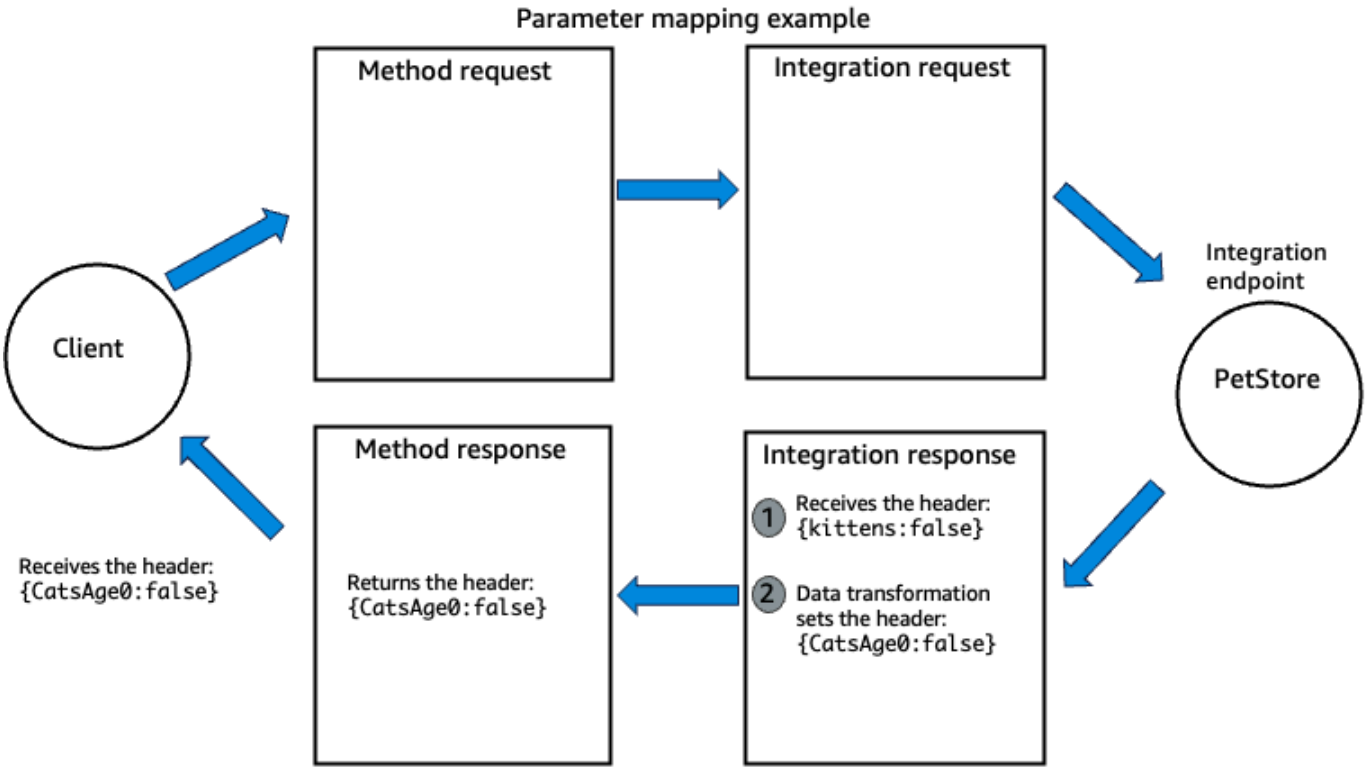
En la asignación de parámetros, asigne parámetros de solicitud o de respuesta. Puede asignar parámetros mediante expresiones de asignación de parámetros o valores estáticos. Para obtener una lista de expresiones de asignación, consulte the section called “Referencia de origen de asignación de parámetros”. Puede usar la asignación de parámetros en la solicitud de integración para integraciones de proxy y de no proxy, pero para usar la asignación de parámetros para una respuesta de integración, necesita una integración de no proxy.

Por ejemplo, puede asignar el parámetro de encabezado de solicitud de método `puppies` al parámetro de encabezado de solicitud de integración `DogsAge0`. A continuación, si un cliente envía el encabezado `puppies:true` a la API, la solicitud de integración envía el encabezado de solicitud `DogsAge0:true` al punto de conexión de integración. En el siguiente diagrama se muestra el ciclo de vida de la solicitud de este ejemplo.



Para crear este ejemplo con API Gateway, consulte the section called “Ejemplo 1: asignar un parámetro de solicitud de método a un parámetro de solicitud de integración”.

Como otro ejemplo, también puede asignar el parámetro de encabezado de respuesta de integración `kittens` al parámetro de encabezado de respuesta de método `CatsAge0`. Si el punto de conexión de integración devuelve `kittens:false`, el cliente recibe el encabezado `CatsAge0:false`. En el siguiente diagrama se muestra el ciclo de vida de la solicitud de este ejemplo.



Temas

- Ejemplos de asignación de parámetros para las API de REST en API Gateway
- Referencia de origen de asignación de parámetros para las API de REST en API Gateway

Ejemplos de asignación de parámetros para las API de REST en API Gateway

En los siguientes ejemplos se muestra cómo crear expresiones de asignación de parámetros mediante la consola de API Gateway, OpenAPI y plantillas de CloudFormation. Para ver un ejemplo de cómo utilizar la asignación de parámetros para crear los encabezados CORS necesarios, consulte the section called “CORS”.

Ejemplo 1: asignar un parámetro de solicitud de método a un parámetro de solicitud de integración

En el siguiente ejemplo se asigna el parámetro de encabezado de solicitud de método `puppies` al parámetro de encabezado de solicitud de integración `DogsAge0`.

Consola de administración de AWS

Asignación del parámetro de solicitud de método

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST.
3. Elija un método.

   El método debe tener una integración sin proxy.
4. En Configuración de solicitud de método, elija Editar.
5. Elija Encabezados de solicitud HTTP.
6. Elija Agregar encabezado.
7. En Nombre, escriba **`puppies`**.
8. Seleccione Save.
9. Elija la pestaña Solicitud de integración y, a continuación, en Configuración de solicitud de integración, elija Editar.

   La Consola de administración de AWS agrega automáticamente una asignación de parámetros de `method.request.header.puppies` a `puppies` por usted, pero debe cambiar el Nombre para que coincida con el parámetro de encabezado de solicitud que espera el punto de conexión de integración.
10. En Nombre, escriba **`DogsAge0`**.
11. Seleccione Save.
12. Vuelva a implementar la API para que los cambios se apliquen.

En los siguientes pasos se muestra cómo verificar que la asignación de parámetros se ha realizado correctamente.

(Opcional) Prueba de la asignación de parámetros

1. Elija la pestaña Prueba. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña.
2. Para los encabezados, introduzca **`puppies:true`**.
3. Seleccione Probar
4. En Registros, el resultado debería tener el siguiente aspecto:

```
Tue Feb 04 00:28:36 UTC 2025 : Method request headers: {puppies=true}
Tue Feb 04 00:28:36 UTC 2025 : Method request body before transformations:
Tue Feb 04 00:28:36 UTC 2025 : Endpoint request URI: http://petstore-demo-
endpoint.execute-api.com/petstore/pets
Tue Feb 04 00:28:36 UTC 2025 : Endpoint request headers: {DogsAge0=true,
 x-amzn-apigateway-api-id=abcd1234, Accept=application/json, User-
Agent=AmazonAPIGateway_aaaaaaa, X-Amzn-Trace-Id=Root=1-abcd-12344}
```

El parámetro de encabezado de solicitud ha cambiado de `puppies` a `DogsAge0`.

CloudFormation

En este ejemplo, utiliza la propiedad body para importar un archivo de definición de OpenAPI en API Gateway.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Body:
        openapi: 3.0.1
        info:
          title: ParameterMappingExample
          version: "2025-02-04T00:30:41Z"
        paths:
          /pets:
            get:
              parameters:
                - name: puppies
                  in: header
                  schema:
                    type: string
              responses:
                "200":
                  description: 200 response
              x-amazon-apigateway-integration:
                httpMethod: GET
                uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets
                responses:
                  default:
```

```
                    statusCode: "200"
                requestParameters:
                  integration.request.header.DogsAge0: method.request.header.puppies
                passthroughBehavior: when_no_match
                type: http
  ApiGatewayDeployment:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn: Api
    Properties:
      RestApiId: !Ref Api
  ApiGatewayDeployment20250219:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn: Api
    Properties:
      RestApiId: !Ref Api
  Stage:
    Type: 'AWS::ApiGateway::Stage'
    Properties:
       DeploymentId: !Ref ApiGatewayDeployment20250219
       RestApiId: !Ref Api
       StageName: prod
```

OpenAPI

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "ParameterMappingExample",
    "version" : "2025-02-04T00:30:41Z"
  },
  "paths" : {
    "/pets" : {
      "get" : {
        "parameters" : [ {
          "name" : "puppies",
          "in" : "header",
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response"
```

```
          }
        },
        "x-amazon-apigateway-integration" : {
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
          "responses" : {
            "default" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.header.DogsAge0" : "method.request.header.puppies"
          },
          "passthroughBehavior" : "when_no_match",
          "type" : "http"
        }
      }
    }
  }
}
```

Ejemplo 2: asignar varios parámetros de solicitud de método a diferentes parámetros de solicitud de integración

En el siguiente ejemplo, se asigna el parámetro de cadena de consulta de solicitud de método de varios valores `methodRequestQueryParam` al parámetro de cadena de consulta de solicitud de integración `integrationQueryParam` y se asigna el parámetro de encabezado de solicitud de método `methodRequestHeaderParam` al parámetro de ruta de solicitud de integración `integrationPathParam`.

Consola de administración de AWS

Asignación de los parámetros de solicitud de método

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST.
3. Elija un método.

   El método debe tener una integración sin proxy.
4. En Configuración de solicitud de método, elija Editar.

5. Elija Parámetros de cadenas de consulta de URL.
6. Elija Añadir cadena de consulta.
7. En Nombre, escriba **`methodRequestQueryParam`**.
8. Elija Encabezados de solicitud HTTP.
9. Elija Agregar encabezado.
10. En Nombre, escriba **`methodRequestHeaderParam`**.
11. Seleccione Save.
12. Elija la pestaña Solicitud de integración y, a continuación, en Configuración de solicitud de integración, elija Editar.
13. Elija los Parámetros de la ruta URL.
14. Elija Añadir parámetro de ruta.
15. En Nombre, escriba **`integrationPathParam`**.
16. En Asignado desde, introduzca **`method.request.header.methodRequestHeaderParam`**.

    Esto asigna el encabezado de solicitud de método que especificó en la solicitud de método a un nuevo parámetro de ruta de solicitud de integración.
17. Elija Parámetros de cadenas de consulta de URL.
18. Elija Añadir cadena de consulta.
19. En Nombre, escriba **`integrationQueryParam`**.
20. En Asignado desde, introduzca **`method.request.multivaluequerystring.methodRequestQueryParam`**.

    Esto asigna el parámetro de cadena de consulta de varios valores a un nuevo parámetro de cadena de consulta de solicitud de integración de valor único.
21. Seleccione Save.
22. Vuelva a implementar la API para que los cambios se apliquen.

CloudFormation

En este ejemplo, utiliza la propiedad body para importar un archivo de definición de OpenAPI en API Gateway.

La siguiente definición de OpenAPI crea las siguientes asignaciones de parámetros para una integración de HTTP:

- El encabezado de la solicitud de método, llamado `methodRequestHeaderParam`, al parámetro de ruta de la solicitud de integración, llamado `integrationPathParam`
- La cadena de consulta de la solicitud de método de varios valores, llamada `methodRequestQueryParam`, a la cadena de consulta de la solicitud de integración, llamada `integrationQueryParam`

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Body:
        openapi: 3.0.1
        info:
          title: Parameter mapping example 2
          version: "2025-01-15T19:12:31Z"
        paths:
          /:
            post:
              parameters:
                - name: methodRequestQueryParam
                  in: query
                  schema:
                    type: string
                - name: methodRequestHeaderParam
                  in: header
                  schema:
                    type: string
              responses:
                "200":
                  description: 200 response
              x-amazon-apigateway-integration:
                httpMethod: GET
                uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets
                responses:
                  default:
                    statusCode: "200"
                requestParameters:
```

```
                  integration.request.querystring.integrationQueryParam:
method.request.multivaluequerystring.methodRequestQueryParam
                  integration.request.path.integrationPathParam:
method.request.header.methodRequestHeaderParam
                requestTemplates:
                  application/json: '{"statusCode": 200}'
                passthroughBehavior: when_no_templates
                timeoutInMillis: 29000
                type: http
  ApiGatewayDeployment:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn: Api
    Properties:
      RestApiId: !Ref Api
  ApiGatewayDeployment20250219:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn: Api
    Properties:
      RestApiId: !Ref Api
  Stage:
    Type: 'AWS::ApiGateway::Stage'
    Properties:
       DeploymentId: !Ref ApiGatewayDeployment20250219
       RestApiId: !Ref Api
       StageName: prod
```

OpenAPI

La siguiente definición de OpenAPI crea las siguientes asignaciones de parámetros para una integración de HTTP:

- El encabezado de la solicitud de método, llamado `methodRequestHeaderParam`, al parámetro de ruta de la solicitud de integración, llamado `integrationPathParam`
- La cadena de consulta de la solicitud de método de varios valores, llamada `methodRequestQueryParam`, a la cadena de consulta de la solicitud de integración, llamada `integrationQueryParam`

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "Parameter mapping example 2",
```

```
    "version" : "2025-01-15T19:12:31Z"
  },
  "paths" : {
    "/" : {
      "post" : {
        "parameters" : [ {
          "name" : "methodRequestQueryParam",
          "in" : "query",
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "methodRequestHeaderParam",
          "in" : "header",
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response"
          }
        },
        "x-amazon-apigateway-integration" : {
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
          "responses" : {
            "default" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.querystring.integrationQueryParam" :
"method.request.multivaluequerystring.methodRequestQueryParam",
            "integration.request.path.integrationPathParam" :
"method.request.header.methodRequestHeaderParam"
          },
          "requestTemplates" : {
            "application/json" : "{\"statusCode\": 200}"
          },
          "passthroughBehavior" : "when_no_templates",
          "timeoutInMillis" : 29000,
          "type" : "http"
        }
```

```
      }
    }
  }
}
```

Ejemplo 3: asignar campos del cuerpo de la solicitud JSON a los parámetros de la solicitud de integración

También puede asignar parámetros de solicitud de integración desde campos del cuerpo de la solicitud JSON mediante una expresión JSONPath. El siguiente ejemplo asigna el cuerpo de solicitud de método a un encabezado de solicitud de integración denominado `body-header` y asigna parte del cuerpo de solicitud, expresado mediante una expresión JSON, a un encabezado de solicitud de integración denominado `pet-price`.

Para probar este ejemplo, proporcione una entrada que contenga una categoría de precio, como la siguiente:

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

Consola de administración de AWS

Asignación de los parámetros de solicitud de método

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST.
3. Elija un método `POST`, `PUT`, `PATCH` o `ANY`.

   El método debe tener una integración sin proxy.
4. En Configuración de solicitud de integración, elija Editar.
5. Elija Parámetros de encabezado de solicitud de URL.
6. Elija Agregar parámetro de encabezado de solicitud.
7. En Nombre, escriba **`body-header`**.

8. En Asignado desde, introduzca **`method.request.body`**.

   Esto asigna el cuerpo de solicitud de método a un nuevo parámetro de encabezado de solicitud de integración.

9. Elija Agregar parámetro de encabezado de solicitud.
10. En Nombre, escriba **`pet-price`**.
11. En Asignado desde, introduzca **`method.request.body[0].price`**.

    Esto asigna una parte del cuerpo de solicitud de método a un nuevo parámetro de encabezado de solicitud de integración.

12. Seleccione Save.
13. Vuelva a implementar la API para que los cambios se apliquen.

CloudFormation

En este ejemplo, utiliza la propiedad body para importar un archivo de definición de OpenAPI en API Gateway.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Body:
        openapi: 3.0.1
        info:
          title: Parameter mapping example 3
          version: "2025-01-15T19:19:14Z"
        paths:
          /:
            post:
              responses:
                "200":
                  description: 200 response
              x-amazon-apigateway-integration:
                httpMethod: GET
                uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets
                responses:
                  default:
                    statusCode: "200"
```

```
                requestParameters:
                  integration.request.header.pet-price: method.request.body[0].price
                  integration.request.header.body-header: method.request.body
                requestTemplates:
                  application/json: '{"statusCode": 200}'
                passthroughBehavior: when_no_templates
                timeoutInMillis: 29000
                type: http
  ApiGatewayDeployment:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn: Api
    Properties:
      RestApiId: !Ref Api
  ApiGatewayDeployment20250219:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn: Api
    Properties:
      RestApiId: !Ref Api
  Stage:
    Type: 'AWS::ApiGateway::Stage'
    Properties:
       DeploymentId: !Ref ApiGatewayDeployment20250219
       RestApiId: !Ref Api
       StageName: prod
```

OpenAPI

La siguiente definición de OpenAPI asigna los parámetros de solicitud de integración de los campos del cuerpo de solicitud JSON.

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "Parameter mapping example 3",
    "version" : "2025-01-15T19:19:14Z"
  },
  "paths" : {
    "/" : {
      "post" : {
        "responses" : {
          "200" : {
            "description" : "200 response"
          }
```

```
        },
        "x-amazon-apigateway-integration" : {
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
          "responses" : {
            "default" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.header.pet-price" : "method.request.body[0].price",
            "integration.request.header.body-header" : "method.request.body"
          },
          "requestTemplates" : {
            "application/json" : "{\"statusCode\": 200}"
          },
          "passthroughBehavior" : "when_no_templates",
          "timeoutInMillis" : 29000,
          "type" : "http"
        }
      }
    }
  }
}
```

Ejemplo 4: asignar la respuesta de integración a la respuesta de método

También puede asignar la respuesta de integración a la respuesta de método. En el siguiente ejemplo se asigna el cuerpo de respuesta de integración a un encabezado de respuesta de método denominado `location`, se asigna el encabezado de respuesta de integración `x-app-id` al encabezado de respuesta de método `id` y se asigna el encabezado de respuesta de integración de varios valores `item` al encabezado de respuesta de método `items`.

Consola de administración de AWS

Asignación de la respuesta de integración

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST.
3. Elija un método.

El método debe tener una integración sin proxy.

4. Elija la pestaña Respuesta de método y, a continuación, para Respuesta 200, elija Editar.
5. En Nombre de encabezado, elija Agregar encabezado.
6. Cree tres encabezados llamados **`id`**, **`item`** y **`location`**.
7. Seleccione Save.
8. Elija la pestaña Respuesta de integración y, a continuación, en Predeterminado: respuesta, elija Editar.
9. En Asignaciones de encabezado, introduzca lo siguiente.
   a. En id, introduzca **`integration.response.header.x-app-id`**
   b. En item, introduzca **`integration.response.multivalueheader.item`**
   c. En location, introduzca **`integration.response.body.redirect.url`**
10. Seleccione Save.
11. Vuelva a implementar la API para que los cambios se apliquen.

CloudFormation

En este ejemplo, utiliza la propiedad body para importar un archivo de definición de OpenAPI en API Gateway.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Body:
        openapi: 3.0.1
        info:
          title: Parameter mapping example
          version: "2025-01-15T19:21:35Z"
        paths:
          /:
            post:
              responses:
                "200":
                  description: 200 response
                  headers:
```

```
                    item:
                      schema:
                        type: string
                    location:
                      schema:
                        type: string
                    id:
                      schema:
                        type: string
              x-amazon-apigateway-integration:
                type: http
                httpMethod: GET
                uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets
                responses:
                  default:
                    statusCode: "200"
                    responseParameters:
                      method.response.header.id: integration.response.header.x-app-
id
                      method.response.header.location:
 integration.response.body.redirect.url
                      method.response.header.item:
 integration.response.multivalueheader.item
                requestTemplates:
                  application/json: '{"statusCode": 200}'
                passthroughBehavior: when_no_templates
                timeoutInMillis: 29000
  ApiGatewayDeployment:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn: Api
    Properties:
      RestApiId: !Ref Api
  ApiGatewayDeployment20250219:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn: Api
    Properties:
      RestApiId: !Ref Api
  Stage:
    Type: 'AWS::ApiGateway::Stage'
    Properties:
       DeploymentId: !Ref ApiGatewayDeployment20250219
       RestApiId: !Ref Api
       StageName: prod
```

OpenAPI

La siguiente definición de OpenAPI asigna la respuesta de integración a la respuesta de método.

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "Parameter mapping example",
    "version" : "2025-01-15T19:21:35Z"
  },
  "paths" : {
    "/" : {
      "post" : {
        "responses" : {
          "200" : {
            "description" : "200 response",
            "headers" : {
              "item" : {
                "schema" : {
                  "type" : "string"
                }
              },
              "location" : {
                "schema" : {
                  "type" : "string"
                }
              },
              "id" : {
                "schema" : {
                  "type" : "string"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration" : {
          "type" : "http",
          "httpMethod" : "GET",
          "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
          "responses" : {
            "default" : {
              "statusCode" : "200",
              "responseParameters" : {
```

```
                "method.response.header.id" : "integration.response.header.x-app-
id",
                "method.response.header.location" :
 "integration.response.body.redirect.url",
                "method.response.header.item" :
 "integration.response.multivalueheader.item"
              }
            }
          },
          "requestTemplates" : {
            "application/json" : "{\"statusCode\": 200}"
          },
          "passthroughBehavior" : "when_no_templates",
          "timeoutInMillis" : 29000
        }
      }
    }
  }
}
```

Referencia de origen de asignación de parámetros para las API de REST en API Gateway

Cuando crea una asignación de parámetro, especifica los parámetros de solicitud de método o de respuesta de integración que desea modificar y especifica cómo modificarlos.

En la siguiente tabla se muestran los parámetros de solicitud de método que puede asignar y la expresión para crear la asignación. En estas expresiones, *name* es el nombre de un parámetro de solicitud de método. Por ejemplo, para asignar el parámetro de encabezado de solicitud `puppies`, utilice la expresión `method.request.header.puppies`. La expresión debe coincidir con la expresión regular `'^[a-zA-Z0-9._$-]+$]'`. Puede utilizar la asignación de parámetros en la solicitud de integración para integraciones de proxy y de no proxy.

Origen de datos asignado	Expresión de asignación
Ruta de solicitud de método	`method.request.path.` *name*
Cadena de consulta de solicitud de método	`method.request.querystring.` *name*
Cadena de consulta de solicitud de método multivalor	`method.request.multivaluequerystring.` *name*

Origen de datos asignado	Expresión de asignación
Encabezado de solicitud de método	`method.request.header.` *`name`*
Encabezado de solicitud de método multivalor	`method.request.multivalueheader.` *`name`*
Cuerpo de solicitud de método	`method.request.body`
Cuerpo de solicitud de método (JsonPath)	`method.request.body.` *`JSONPath_EXPRESSION`* . *`JSONPath_EXPRESSION`* es una expresión JSONPath para un campo JSON del cuerpo de una solicitud. Para obtener más información, consulte Expresión JSONPath.
Variables de etapa	`stageVariables.` *`name`*
Variables de contexto	`context.`*`name`* El nombre debe ser una de las variables de contexto admitidas.
Valor estático	*`'static_value'`* . El valor de *`static_value`* es un literal de cadena que se debe incluir entre comillas simples. Por ejemplo, `'https://www.example.com'` .

En la siguiente tabla se muestran los parámetros de respuesta de integración que puede asignar y la expresión para crear la asignación. En estas expresiones, *`name`* es el nombre de un parámetro de respuesta de integración. Puede asignar encabezados de respuesta de método desde cualquier encabezado de respuesta de integración o cuerpo de respuesta de integración, variables de contexto o valores estáticos. Para utilizar la asignación de parámetros para una respuesta de integración, necesita una integración de no proxy.

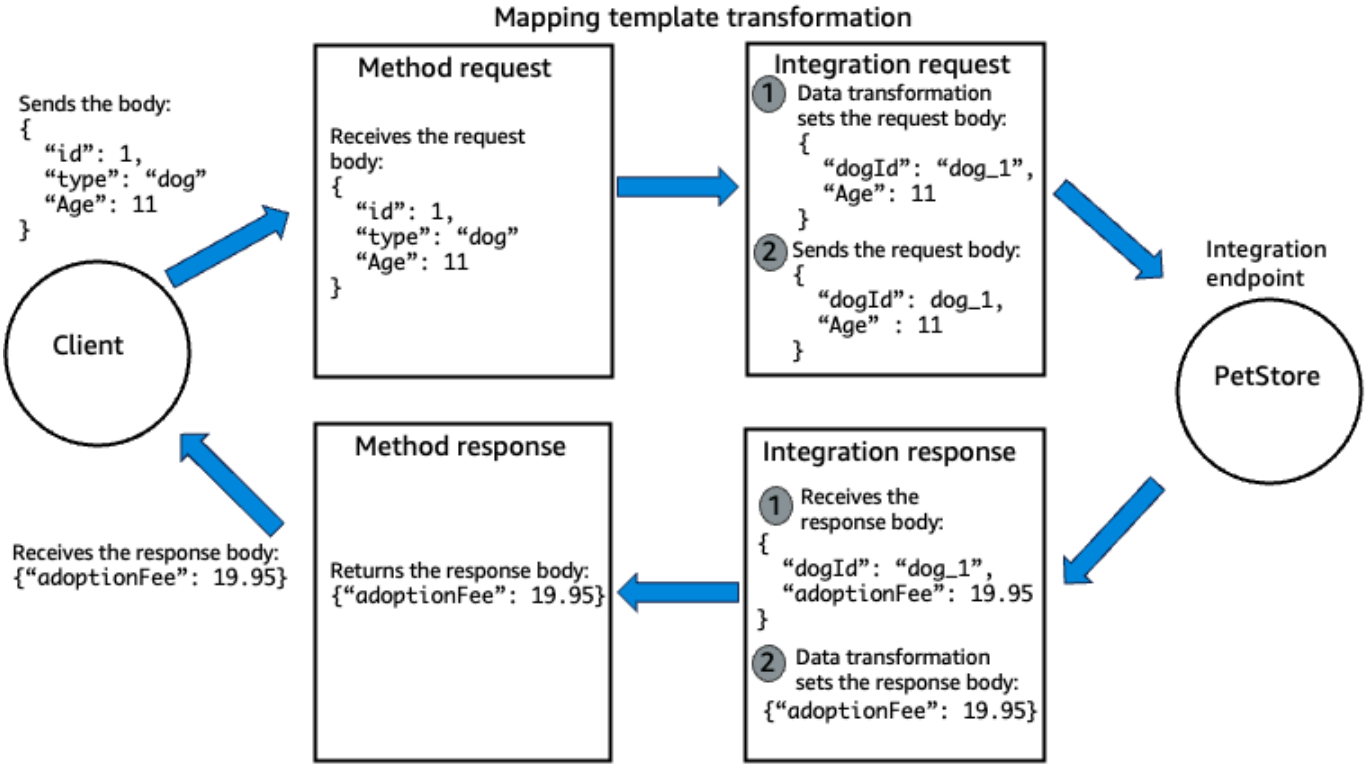
Origen de datos asignado	Expresión de asignación
Encabezado de respuesta de integración	`integration.response.header.` *`name`*
Encabezado de respuesta de integración	`integration.response.multivalueheader.` *`name`*
Cuerpo de respuesta de integración	`integration.response.body`
Cuerpo de respuesta de integración (JsonPath)	`integration.response.body.` *`JSONPath_EXPRESSION`* *`JSONPath_EXPRESSION`* es una expresión JSONPath para un campo JSON del cuerpo de una respuesta. Para obtener más información, consulte Expresión JSONPath.
Variable de etapa	`stageVariables.` *`name`*
Variable de contexto	`context.`*`name`* El nombre debe ser una de las variables de contexto admitidas.
Valor estático	*`'static_value'`* El valor de *`static_value`* es un literal de cadena que se debe incluir entre comillas simples. Por ejemplo, `'https://www.example.com'` .

Asignación de las transformaciones de plantillas de asignación para las API de REST en API Gateway

Las transformaciones de plantillas de asignación utilizan una plantilla de asignación para modificar la solicitud de integración o respuesta de integración. Una plantilla de asignación es un script expresado en Velocity Template Language (VTL) y aplicado a una carga útil mediante JSONPath basado en el encabezado `Content-type`. Utiliza plantillas de asignación cuando utiliza las

transformaciones de plantillas de asignación. En esta sección se describe la información conceptual relacionada con las plantillas de asignación.

En el siguiente diagrama se muestra el ciclo de vida de la solicitud de un recurso `POST /pets` que tiene una integración con un punto de conexión de integración de PetStore. En esta API, un usuario envía datos sobre una mascota y el punto de conexión de integración devuelve la tarifa de adopción asociada a una mascota. En este ciclo de vida de la solicitud, las transformaciones de plantillas de asignación filtran el cuerpo de la solicitud al punto de conexión de integración y filtran el cuerpo de la respuesta desde el punto de conexión de integración.



En esta sección se explica el ciclo de vida de la solicitud y la respuesta.

Solicitud de método y solicitud de integración

En el ejemplo anterior, si este es el cuerpo de solicitud enviado al método de solicitud:

```
POST /pets
    HTTP/1.1
    Host:abcd1234.us-west-2.amazonaws.com
    Content-type: application/json

  {
```

```
    "id": 1,
    "type": "dog",
    "Age": 11,
  }
```

Este cuerpo de solicitud no tiene el formato correcto para que lo utilice el punto de conexión de integración, por lo que API Gateway realiza las transformaciones de plantilla de asignación. API Gateway solo realiza las transformaciones de plantilla de asignación porque hay una plantilla de asignación definida para Content-Type `application/json`. Si no define una plantilla de asignación para Content-Type, de forma predeterminada, API Gateway pasa el cuerpo a través de la solicitud de integración al punto de conexión de integración. Para modificar este comportamiento, consulte the section called “Comportamiento de solicitud de método para cargas útiles sin plantillas de asignación”.

La siguiente plantilla de asignación transforma los datos de solicitud de método en la solicitud de integración antes de que se envíen al punto de conexión de integración:

```
#set($inputRoot = $input.path('$'))
  {
    "dogId" : "dog_"$elem.id,
    "Age": $inputRoot.Age
  }
```

1. La variable `$inputRoot` representa el objeto raíz en los datos JSON originales de la sección anterior. Las directivas comienzan con el símbolo #.
2. El valor `dog` es una concatenación del `id` del usuario y un valor de cadena.
3. `Age` es del cuerpo de solicitud de método.

A continuación, se reenvía el siguiente resultado al punto de conexión de integración:

```
{
    "dogId" : "dog_1",
    "Age": 11
  }
```

Respuesta de integración y respuesta de método

Después de que la solicitud se haya enviado correctamente al punto de conexión de integración, el punto de conexión envía una respuesta a la respuesta de integración de API Gateway. A continuación, se muestran los datos de salida de ejemplo del punto de conexión de integración:

```
{
    "dogId" : "dog_1",
    "adoptionFee": 19.95,
}
```

La respuesta de método espera una carga útil diferente a la que devuelve la respuesta de integración. API Gateway realiza las transformaciones de plantillas de asignación. API Gateway solo realiza las transformaciones de plantilla de asignación porque hay una plantilla de asignación definida para Content-Type `application/json`. Si no define una plantilla de asignación para Content-Type, de forma predeterminada, API Gateway pasa el cuerpo a través de la respuesta de integración a la respuesta de método. Para modificar este comportamiento, consulte the section called “Comportamiento de solicitud de método para cargas útiles sin plantillas de asignación”.

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

El siguiente resultado se envía a la respuesta del método:

```
{"adoptionFee": 19.95}
```

Esto completa el ejemplo de transformación de plantilla de asignación. Recomendamos que, cuando sea posible, en lugar de utilizar las transformaciones de plantilla de asignación, utilice una integración de proxy para transformar los datos. Para obtener más información, consulte the section called “Selección de un tipo de integración de la API”.

Comportamiento de solicitud de método para cargas útiles sin plantillas de asignación para las API de REST en API Gateway

Si la solicitud de método tiene una carga útil y no tiene una plantilla de asignación definida para el encabezado `Content-Type`, puede elegir pasar la carga útil de la solicitud suministrada por el cliente a través de la solicitud de integración al backend sin transformación. Este proceso se denomina "acceso directo a la integración".

El comportamiento de acceso directo real de una solicitud entrante viene determinado por esta configuración. Hay tres opciones:

Cuando ninguna plantilla coincide con el encabezado Tipo de contenido solicitado

Elija esta opción si desea que el cuerpo de la solicitud de método se transmita a través de la solicitud de integración al backend sin transformación cuando el tipo de contenido de la solicitud de método no coincida con ningún tipo de contenido asociado a las plantillas de mapeo.

Al llamar a la API de API Gateway, se elige esta opción estableciendo `WHEN_NO_MATCH` como el valor de la propiedad `passthroughBehavior` en Integración.

Cuando no hay plantillas definidas (recomendado)

Elija esta opción si desea que el cuerpo de la solicitud de método se transmita a través de la solicitud de integración al backend sin transformación cuando no se haya definido ninguna plantilla de mapeo en la solicitud de integración. Si se define una plantilla cuando se selecciona esta opción, la solicitud de método con una carga útil y un tipo de contenido que no coincida con ninguna plantilla de asignación definida se rechazará con una respuesta HTTP 415 Tipo de medio no admitido.

Al llamar a la API de API Gateway, se elige esta opción estableciendo `WHEN_NO_TEMPLATES` como el valor de la propiedad `passthroughBehavior` en Integración.

Nunca

Elija esta opción si no desea que el cuerpo de la solicitud de método se transmita a través de la solicitud de integración al backend sin transformación cuando no se haya definido ninguna plantilla de mapeo en la solicitud de integración. Si se define una plantilla cuando esta opción está seleccionada, la solicitud de método de un tipo de contenido sin asignar se rechazará con una respuesta HTTP 415 Unsupported Media Type.

Al llamar a la API de API Gateway, se elige esta opción estableciendo `NEVER` como el valor de la propiedad `passthroughBehavior` en Integración.

Los siguientes ejemplos muestran los posibles comportamientos del acceso directo.

Ejemplo 1: Se define una plantilla de asignación en la solicitud de integración para el tipo de contenido `application/json`.

content-type	Opción de acceso directo	Comportamiento
Ninguno El valor predeterminado de API Gateway es `application/json`	`WHEN_NO_MATCH`	La carga de solicitud se transforma mediante la plantilla.
Ninguno El valor predeterminado de API Gateway es `application/json`	`WHEN_NO_TEMPLATES`	La carga de solicitud se transforma mediante la plantilla.
Ninguno El valor predeterminado de API Gateway es `application/json`	`NEVER`	La carga de solicitud se transforma mediante la plantilla.
`application/json`	`WHEN_NO_MATCH`	La carga de solicitud se transforma mediante la plantilla.
`application/json`	`WHEN_NO_TEMPLATES`	La carga de solicitud se transforma mediante la plantilla.
`application/json`	`NEVER`	La carga de solicitud se transforma mediante la plantilla.
`application/xml`	`WHEN_NO_MATCH`	La carga de solicitud no se transforma y se envía al backend tal como está.
`application/xml`	`WHEN_NO_TEMPLATES`	La solicitud se rechaza con una respuesta HTTP 415

content-type	Opción de acceso directo	Comportamiento
		`Unsupported Media Type`.
`application/xml`	`NEVER`	La solicitud se rechaza con una respuesta HTTP `415 Unsupported Media Type`.

Ejemplo 2: Se define una plantilla de asignación en la solicitud de integración para el tipo de contenido `application/xml`.

Content-type	Opción de acceso directo	Comportamiento
Ninguno El valor predeterminado de API Gateway es `application/json`	`WHEN_NO_MATCH`	La carga de solicitud no se transforma y se envía al backend tal como está.
Ninguno El valor predeterminado de API Gateway es `application/json`	`WHEN_NO_TEMPLATES`	La solicitud se rechaza con una respuesta HTTP `415 Unsupported Media Type`.
Ninguno El valor predeterminado de API Gateway es `application/json`	`NEVER`	La solicitud se rechaza con una respuesta HTTP `415 Unsupported Media Type`.
`application/json`	`WHEN_NO_MATCH`	La carga de solicitud no se transforma y se envía al backend tal como está.
`application/json`	`WHEN_NO_TEMPLATES`	La solicitud se rechaza con una respuesta HTTP `415`

Content-type	Opción de acceso directo	Comportamiento
		`Unsupported Media Type`.
`application/json`	`NEVER`	La solicitud se rechaza con una respuesta HTTP `415 Unsupported Media Type`.
`application/xml`	`WHEN_NO_MATCH`	La carga de solicitud se transforma mediante la plantilla.
`application/xml`	`WHEN_NO_TEMPLATES`	La carga de solicitud se transforma mediante la plantilla.
`application/xml`	`NEVER`	La carga de solicitud se transforma mediante la plantilla.

Ejemplo 3: no se han definido plantillas de asignación en la solicitud de integración.

Content-type	Opción de acceso directo	Comportamiento
Ninguno El valor predeterminado de API Gateway es `application/json`	`WHEN_NO_MATCH`	La carga de solicitud no se transforma y se envía al backend tal como está.
Ninguno El valor predeterminado de API Gateway es `application/json`	`WHEN_NO_TEMPLATES`	La carga de solicitud no se transforma y se envía al backend tal como está.

Content-type	Opción de acceso directo	Comportamiento
Ninguno El valor predeterminado de API Gateway es `application/json`	`NEVER`	La solicitud se rechaza con una respuesta HTTP `415 Unsupported Media Type`.
`application/json`	`WHEN_NO_MATCH`	La carga de solicitud no se transforma y se envía al backend tal como está.
`application/json`	`WHEN_NO_TEMPLATES`	La carga de solicitud no se transforma y se envía al backend tal como está.
`application/json`	`NEVER`	La solicitud se rechaza con una respuesta HTTP `415 Unsupported Media Type`.
`application/xml`	`WHEN_NO_MATCH`	La carga de solicitud no se transforma y se envía al backend tal como está.
`application/xml`	`WHEN_NO_TEMPLATES`	La carga de solicitud no se transforma y se envía al backend tal como está.
`application/xml`	`NEVER`	La solicitud se rechaza con una respuesta HTTP `415 Unsupported Media Type`.

Ejemplo de plantilla de asignación adicional para las API de REST en API Gateway

En el siguiente ejemplo se muestra una API de álbum de fotos en API Gateway que utiliza plantillas de asignación para transformar los datos de solicitud de integración y respuesta de integración.

También utiliza modelos de datos para definir la carga útil de las solicitudes de métodos y las respuestas de integración. Para obtener más información acerca de los modelos de datos, consulte Modelos de datos para las API de REST.

Solicitud de método y solicitud de integración

A continuación se muestra un modelo que define el cuerpo de solicitud de método. Este modelo de entrada requiere que el intermediario cargue una página de fotos y un mínimo de 10 fotos por cada página. Puede utilizar este modelo de entrada para generar un SDK o para utilizar la validación de solicitudes para la API. Durante la validación de la solicitud, si el cuerpo de solicitud de método no se adhiere a la estructura de datos del modelo, API Gateway rechaza la solicitud.

```
{
  "$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" }
```

```
            }
          }
        }
      }
    }
  }
}
```

A continuación, se muestra un ejemplo de cuerpo de solicitud de método que se adhiere a la estructura de datos del modelo de datos anterior.

```
{
  "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
```

```
      }
    ]
  }
}
```

En este ejemplo, si el cliente ha enviado el cuerpo de solicitud de método anterior, esta plantilla de asignación transforma la carga útil para que coincida con el formato requerido por el punto de conexión de integración.

```
#set($inputRoot = $input.path('$'))
{
  "photos": [
#foreach($elem in $inputRoot.photos.photo)
    {
      "id": "$elem.id",
      "photographedBy": "$elem.photographer_first_name $elem.photographer_last_name",
      "title": "$elem.title",
      "ispublic": $elem.ispublic,
      "isfriend": $elem.isfriend,
      "isfamily": $elem.isfamily
    }#if($foreach.hasNext),#end

#end
  ]
}
```

En el siguiente ejemplo se muestran los datos de salida de la transformación:

```
{
  "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
    }
  ]
}
```

Estos datos se envían a la solicitud de integración y, a continuación, al punto de conexión de integración.

Respuesta de integración y respuesta de método

A continuación, se muestra un modelo de salida de ejemplo para los datos fotográficos del punto de conexión de integración. Puede usar este modelo como modelo de respuesta de método, que es necesario al generar un SDK con establecimiento inflexible de tipos para la API. Esto provoca que el resultado se traduzca a una clase apropiada en Java u Objective-C.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "PhotosOutputModel",
  "type": "object",
  "properties": {
    "photos": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "id": { "type": "string" },
          "photographedBy": { "type": "string" },
          "title": { "type": "string" },
          "ispublic": { "type": "boolean" },
          "isfriend": { "type": "boolean" },
          "isfamily": { "type": "boolean" }
        }
      }
    }
  }
}
```

El punto de conexión de integración podría no responder con una respuesta que se adhiera a la estructura de datos de este modelo. Por ejemplo, la respuesta de integración podría tener el siguiente aspecto:

```
  "photos": [
    {
      "id": "12345678901",
      "photographedBy": "Saanvi Sarkar",
      "title": "Sample photo 1",
      "description": "My sample photo 1",
      "public": true,
      "friend": false,
      "family": false
    },
    {
      "id": "23456789012",
      "photographedBy": "Richard Roe",
      "title": "Sample photo 2",
      "description": "My sample photo 1",
      "public": true,
      "friend": false,
      "family": false
    }
  ]
}
```

El siguiente ejemplo de plantilla de asignación transforma los datos de respuesta de integración al formato esperado por la respuesta de método:

```
#set($inputRoot = $input.path('$'))
{
  "photos": [
#foreach($elem in $inputRoot.photos.photo)
    {
      "id": "$elem.id",
      "photographedBy": "$elem.photographer_first_name $elem.photographer_last_name",
      "title": "$elem.title",
      "ispublic": $elem.public,
      "isfriend": $elem.friend,
      "isfamily": $elem.family
    }#if($foreach.hasNext),#end

#end
  ]
}
```

En el siguiente ejemplo se muestran los datos de salida de la transformación:

```
{
  "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
    }
  ]
}
```

Estos datos se envían a la respuesta del método y luego se devuelven al cliente.

Anulación de los parámetros de solicitud y respuesta y de los códigos de estado de la API para las API de REST en API Gateway

Puede utilizar las transformaciones de plantilla de asignación para anular cualquier tipo de parámetro de solicitud, encabezado de respuesta o código de estado de respuesta. Utiliza una plantilla de asignación para hacer lo siguiente:

- Realizar asignaciones de parámetros muchos a uno
- Anular parámetros una vez aplicadas las asignaciones de API Gateway estándar
- Asignar parámetros condicionalmente en función del contenido del cuerpo u otros valores de parámetro
- Crear nuevos parámetros mediante programación
- Anular los códigos de estado devueltos por el punto de conexión de integración

Las invalidaciones son definitivas. Una invalidación solo puede aplicarse a cada parámetro una vez. Si intenta anular el mismo parámetro varias veces, API Gateway devuelve una respuesta 5XX. Si tiene que invalidar el mismo parámetro varias veces en la plantilla, le recomendamos crear una variable y aplicar la invalidación al final de la plantilla. La plantilla solo se aplica una vez analizada en su totalidad.

Ejemplo 1: anular el código de estado en función del cuerpo de integración

En el siguiente ejemplo se utiliza la API de ejemplo para anular el código de estado en función del cuerpo de la respuesta de integración.

Consola de administración de AWS

Anulación de un código de estado basado en el cuerpo de respuesta de integración

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Seleccione Crear API.
3. En API de REST, elija Compilación.
4. En Detalles de la API, elija API de ejemplo.
5. Seleccione Crear API.

   API Gateway crea una API de tienda de mascotas de ejemplo. Para recuperar información sobre una mascota, utilice la solicitud de método de API de `GET /pets/{petId}`, donde `{petId}` es un parámetro de ruta correspondiente a un número de identificación de una mascota.

   En este ejemplo, anula el código de respuesta del método `GET` a `400` cuando se detecta una condición de error.
6. En el árbol Recursos, elija el método `GET` en `/{petId}`.
7. Primero, pruebe la implementación actual de la API.

   Elija la pestaña Prueba. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña.
8. En petId, introduzca **`-1`** y, a continuación, seleccione Prueba.

   El Cuerpo de respuesta indica un error de fuera de intervalo:

```
{
  "errors": [
```

```
    {
      "key": "GetPetRequest.petId",
      "message": "The value is out of range."
    }
  ]
}
```

Además, la última línea bajo Registros termina por `Method completed with status: 200`.

La integración se ha completado correctamente, pero se ha producido un error. Ahora, anulará el código de estado basado en el cuerpo de respuesta de integración.

9. En la pestaña Respuesta de integración, en Predeterminado: respuesta, seleccione Editar.
10. Elija Plantillas de mapeo.
11. Elija Add mapping template (Añadir plantilla de asignación).
12. En Tipo de contenido, ingrese **`application/json`**.
13. En Cuerpo de la plantilla, introduzca lo siguiente:

```
#set($inputRoot = $input.path('$'))
$input.json("$")
#if($inputRoot.toString().contains("error"))
#set($context.responseOverride.status = 400)
#end
```

Esta plantilla de asignación utiliza la variable `$context.responseOverride.status` para anular el código de estado a `400` si la respuesta de integración contiene la cadena `error`.

14. Seleccione Save.
15. Elija la pestaña Prueba.
16. En petId, introduzca **`-1`**.
17. En los resultados, el Response Body (Cuerpo de respuesta) indica un error de fuera de intervalo:

```
{
  "errors": [
    {
      "key": "GetPetRequest.petId",
```

```
      "message": "The value is out of range."
    }
  ]
}
```

Sin embargo, la última línea de Registros termina ahora por `Method completed with status: 400`.

CloudFormation

En este ejemplo, utiliza la propiedad body para importar un archivo de definición de OpenAPI en API Gateway.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Body:
        openapi: 3.0.1
        info:
          title: PetStore Example 1
          description: Example pet store API.
          version: "2025-01-14T00:13:18Z"
        paths:
          /pets/{petId}:
            get:
              parameters:
                - name: petId
                  in: path
                  required: true
                  schema:
                    type: string
              responses:
                "200":
                  description: 200 response
              x-amazon-apigateway-integration:
                httpMethod: GET
                uri: http://petstore.execute-api.us-east-1.amazonaws.com/petstore/
pets/{petId}
                responses:
                  default:
```

```
                    statusCode: "200"
                    responseTemplates:
                      application/json: |-
                        #set($inputRoot = $input.path('$'))
                        $input.json("$")
                        #if($inputRoot.toString().contains("error"))
                        #set($context.responseOverride.status = 400)
                        #end
                requestParameters:
                  integration.request.path.petId: method.request.path.petId
                passthroughBehavior: when_no_match
                type: http
        components:
          schemas:
            Pet:
              type: object
              properties:
                id:
                  type: integer
                type:
                  type: string
                price:
                  type: number
  ApiGatewayDeployment:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn: Api
    Properties:
      RestApiId: !Ref Api
  ApiGatewayDeployment20250219:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn: Api
    Properties:
      RestApiId: !Ref Api
  Stage:
    Type: 'AWS::ApiGateway::Stage'
    Properties:
       DeploymentId: !Ref ApiGatewayDeployment20250219
       RestApiId: !Ref Api
       StageName: prod
```

OpenAPI

La siguiente definición de OpenAPI crea el recurso `GET pets/{petId}` y anula el código de estado en función del cuerpo de integración.

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "PetStore Example 1",
    "description" : "Example pet store API.",
    "version" : "2025-01-14T00:13:18Z"
  },
  "paths" : {
    "/pets/{petId}" : {
      "get" : {
        "parameters" : [ {
          "name" : "petId",
          "in" : "path",
          "required" : true,
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response"
          }
        },
        "x-amazon-apigateway-integration" : {
          "httpMethod" : "GET",
          "uri" : "http://petstore.execute-api.us-east-1.amazonaws.com/petstore/
pets/{petId}",
          "responses" : {
            "default" : {
              "statusCode" : "200",
              "responseTemplates" : {
                "application/json" : "#set($inputRoot = $input.path('$'))\n
$input.json(\"$\")\n#if($inputRoot.toString().contains(\"error
\"))\n#set($context.responseOverride.status = 400)\n#end"
              }
            }
          },
          "requestParameters" : {
            "integration.request.path.petId" : "method.request.path.petId"
          },
          "passthroughBehavior" : "when_no_match",
          "type" : "http"
        }
```

```
      }
    }
  },
  "components" : {
    "schemas" : {
      "Pet" : {
        "type" : "object",
        "properties" : {
          "id" : {
            "type" : "integer"
          },
          "type" : {
            "type" : "string"
          },
          "price" : {
            "type" : "number"
          }
        }
      }
    }
  }
}
```

Ejemplo 2: anular el encabezado de solicitud y crear nuevos encabezados

En el siguiente ejemplo se utiliza la API de ejemplo para anular el encabezado de solicitud y crear nuevos encabezados.

Consola de administración de AWS

Anulación del encabezado de solicitud de un método mediante la creación de un nuevo encabezado

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Elija la API de ejemplo que creó en el tutorial anterior. El nombre de la API debe ser PetStore.
3. En el árbol Recursos, elija el método `GET` en `/pet`.
4. En la pestaña Solicitud de método, en Configuración de solicitud de método, elija Editar.
5. Elija Encabezados de solicitud HTTP y, a continuación, elija Agregar encabezado.
6. En Nombre, escriba **`header1`**.

7. Elija Agregar encabezado y, a continuación, cree un segundo encabezado llamado **`header2`**.
8. Seleccione Save.

   Ahora, combina estos encabezados en un valor de encabezado mediante una plantilla de asignación.
9. En la pestaña Solicitud de integración, en Configuración de la solicitud de integración, seleccione Editar.
10. En Acceso directo de cuerpo de la solicitud, elija Cuando no haya plantillas definidas (recomendado).
11. Elija Plantillas de mapeo y, a continuación, haga lo siguiente:

    a. Elija Add mapping template (Añadir plantilla de asignación).

    b. En Tipo de contenido, ingrese **`application/json`**.

    c. En Cuerpo de la plantilla, introduzca lo siguiente:

    ```
    #set($header1Override = "pets")
    #set($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])
    ```

    Esta plantilla de asignación anula `header1` con la cadena `pets` y crea un encabezado de varios valores llamado `$header3Value` que combina `header1` y `header2`.
12. Seleccione Save.
13. Elija la pestaña Prueba.
14. En Encabezados, copie el siguiente código:

    ```
    header1:header1Val
    header2:header2Val
    ```
15. Seleccione Test (Probar).

    En el cuadro Registros, debería ver una entrada que incluye este texto:

    ```
    Endpoint request headers: {header3=header1Valheader2Val,
    header2=header2Val, header1=pets, x-amzn-apigateway-api-id=api-id,
    ```

```
Accept=application/json, multivalueheader=pets,header1Valheader2Val}
```

CloudFormation

En este ejemplo, utiliza la propiedad body para importar un archivo de definición de OpenAPI en API Gateway.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Body:
        openapi: 3.0.1
        info:
          title: PetStore Example 2
          description: Example pet store API.
          version: "2025-01-14T00:36:18Z"
        paths:
          /pets:
            get:
              parameters:
                - name: header2
                  in: header
                  schema:
                    type: string
                - name: page
                  in: query
                  schema:
                    type: string
                - name: type
                  in: query
                  schema:
                    type: string
                - name: header1
                  in: header
                  schema:
                    type: string
              responses:
                "200":
                  description: 200 response
              x-amazon-apigateway-integration:
```

```
                httpMethod: GET
                uri: http://petstore.execute-api.us-east-1.amazonaws.com/petstore/
pets
                responses:
                  default:
                    statusCode: "200"
                requestParameters:
                  integration.request.header.header1: method.request.header.header1
                  integration.request.header.header2: method.request.header.header2
                  integration.request.querystring.page:
 method.request.querystring.page
                  integration.request.querystring.type:
 method.request.querystring.type
                requestTemplates:
                  application/json: |-
                    #set($header1Override = "pets")
                    #set($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])
                passthroughBehavior: when_no_match
                type: http
        components:
          schemas:
            Pet:
              type: object
              properties:
                id:
                  type: integer
                type:
                  type: string
                price:
                  type: number
  ApiGatewayDeployment:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn: Api
    Properties:
      RestApiId: !Ref Api
  ApiGatewayDeployment20250219:
    Type: 'AWS::ApiGateway::Deployment'
```

```
    DependsOn: Api
    Properties:
      RestApiId: !Ref Api
  Stage:
    Type: 'AWS::ApiGateway::Stage'
    Properties:
       DeploymentId: !Ref ApiGatewayDeployment20250219
       RestApiId: !Ref Api
       StageName: prod
```

OpenAPI

La siguiente definición de OpenAPI crea el recurso `GET pets` y anula el encabezado de solicitud y crea nuevos encabezados.

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "PetStore Example 2",
    "description" : "Example pet store API.",
    "version" : "2025-01-14T00:36:18Z"
  },
  "paths" : {
    "/pets" : {
      "get" : {
        "parameters" : [ {
          "name" : "header2",
          "in" : "header",
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "page",
          "in" : "query",
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "type",
          "in" : "query",
          "schema" : {
            "type" : "string"
          }
```

```
        }, {
          "name" : "header1",
          "in" : "header",
          "schema" : {
            "type" : "string"
          }
        } ],
        "responses" : {
          "200" : {
            "description" : "200 response"
          }
        },
        "x-amazon-apigateway-integration" : {
          "httpMethod" : "GET",
          "uri" : "http://petstore.execute-api.us-east-1.amazonaws.com/petstore/
pets",
          "responses" : {
            "default" : {
              "statusCode" : "200"
            }
          },
          "requestParameters" : {
            "integration.request.header.header1" : "method.request.header.header1",
            "integration.request.header.header2" : "method.request.header.header2",
            "integration.request.querystring.page" :
 "method.request.querystring.page",
            "integration.request.querystring.type" :
 "method.request.querystring.type"
          },
          "requestTemplates" : {
            "application/json" : "#set($header1Override = \"pets
\")\n#set($header3Value = \"$input.params('header1')$input.params('header2')\")\n
$input.json(\"$\")\n#set($context.requestOverride.header.header3
 = $header3Value)\n#set($context.requestOverride.header.header1 =
 $header1Override)\n#set($context.requestOverride.header.multivalueheader=[$header1Override,
 $header3Value])"
          },
          "passthroughBehavior" : "when_no_match",
          "type" : "http"
        }
      }
    }
  }
```

```
}
```

Para utilizar una anulación de plantilla de asignación, agregue una o más de las siguientes variables `$context`. Para ver una lista de variables `$context`, consulte the section called “Variables de contexto para transformaciones de datos”.

Tutorial: modificación de la solicitud y respuesta de integración para integraciones con servicios de AWS

En el siguiente tutorial se muestra cómo utilizar las transformaciones de plantilla de asignación para configurar plantillas de asignación con el fin de transformar las solicitudes y respuestas de integración mediante la consola y la CLI de AWS.

Temas

- Configuración de la transformación de datos en la consola de API Gateway
- Configuración de la transformación de datos mediante la CLI de AWS
- Plantilla CloudFormation de transformación de datos completada

Configuración de la transformación de datos en la consola de API Gateway

En este tutorial, creará una API y una tabla de DynamoDB incompletas con el siguiente archivo .zip data-transformation-tutorial-console.zip. Esta API incompleta tiene un recurso `/pets` con los métodos `GET` y `POST`.

- El método `GET` obtendrá datos del punto de conexión HTTP `http://petstore-demo-endpoint.execute-api.com/petstore/pets`. Los datos de salida se transformarán según la plantilla de mapeo en the section called “Transformaciones de plantillas de asignación”.
- El método `POST` permitirá al usuario `POST` información de la mascota en una tabla de Amazon DynamoDB mediante una plantilla de mapeo.

Descargue y descomprima la plantilla de creación de aplicaciones para CloudFormation. Utilizará esta plantilla para crear una tabla de DynamoDB para publicar información sobre la mascota y una API incompleta. Finalizará el resto de los pasos en la consola de API Gateway.

Para crear una pila de CloudFormation

1. Abra la consola de CloudFormation en https://console.aws.amazon.com/cloudformation.

2. Seleccione Create stack (Crear pila) y, a continuación, seleccione With new resources (standard) (Con nuevos recursos [estándar]).
3. En Specify template (Especificar plantilla), elija Upload a template file (Cargar un archivo de plantilla).
4. Seleccione la plantilla que ha descargado.
5. Elija Next (Siguiente).
6. En Stack name (Nombre de pila), escriba **`data-transformation-tutorial-console`** y, a continuación, elija Next (Siguiente).
7. En Configure stack options (Configurar opciones de pila), elija Next (Siguiente).
8. Para Capabilities (Capacidades), sepa que CloudFormation puede crear recursos de IAM en su cuenta.
9. Elija Siguiente y, a continuación, elija Enviar.

CloudFormation aprovisiona los recursos especificados en la plantilla. Puede tardar varios minutos en finalizar el aprovisionamiento de los recursos. Cuando el estado de la pila de CloudFormation sea CREATE_COMPLETE, estará listo para continuar con el paso siguiente.

Para probar la respuesta de integración de **`GET`**

1. En la pestaña Recursos de la pila CloudFormation de **`data-transformation-tutorial-console`**, seleccione el ID físico de su API.
2. En el panel de navegación principal, seleccione Recursos y, a continuación, seleccione el método GET.
3. Elija la pestaña Prueba. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña.

   El resultado de la prueba mostrará lo siguiente:

```
[
  {
    "id": 1,
    "type": "dog",
    "price": 249.99
  },
  {
    "id": 2,
    "type": "cat",
```

```
    "price": 124.99
  },
  {
    "id": 3,
    "type": "fish",
    "price": 0.99
  }
]
```

Transformará este resultado según la plantilla de mapeo en the section called “Transformaciones de plantillas de asignación”.

Para transformar la respuesta de integración de **GET**

1. Elija la pestaña Respuesta de integración.

   Actualmente, no hay plantillas de mapeo definidas, por lo que la respuesta de integración no se transformará.
2. En Predeterminado: respuesta, seleccione Editar.
3. Elija Plantillas de mapeo y, a continuación, haga lo siguiente:

   a. Elija Add mapping template (Añadir plantilla de asignación).

   b. En Tipo de contenido, ingrese **application/json**.

   c. En Cuerpo de la plantilla, introduzca lo siguiente:

```
#set($inputRoot = $input.path('$'))
[
#foreach($elem in $inputRoot)
  {
    "description" : "Item $elem.id is a $elem.type.",
    "askingPrice" : $elem.price
  }#if($foreach.hasNext),#end

#end
]
```

   Seleccione Save.

Para probar la respuesta de integración de **GET**

- Elija la pestaña Prueba y, a continuación, seleccione Prueba.

  El resultado de la prueba mostrará la respuesta transformada.

```
[
  {
    "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
  }
]
```

Para transformar los datos de entrada del método **POST**

1. Elija el método POST.
2. Elija la pestaña Solicitud de integración y, a continuación, en Configuración de solicitud de integración, elija Editar.

   La plantilla CloudFormation ha rellenado algunos de los campos de la solicitud de integración.

   - El tipo de integración es Servicio de AWS.
   - El Servicio de AWS es DynamoDB.
   - El método HTTP es `POST`.
   - La acción es `PutItem`.
   - La función de ejecución que permite a API Gateway incluir un elemento en la tabla de DynamoDB es `data-transformation-tutorial-console-APIGatewayRole`. CloudFormation ha creado este rol para permitir que API Gateway tuviera los permisos mínimos para interactuar con DynamoDB.

No se ha especificado el nombre de la tabla de DynamoDB. Especificará el nombre en los pasos siguientes.

3. En Acceso directo de cuerpo de la solicitud, seleccione Nunca.

   Esto significa que la API rechazará los datos con tipos de contenido que no tengan una plantilla de mapeo.

4. Elija Plantillas de mapeo.

5. El Tipo de contenido está establecido en `application/json`. Esto significa que la API rechazará un tipo de contenido que no sea application/json. Para obtener más información sobre los comportamientos del acceso directo a la integración, consulte . Comportamiento de solicitud de método para cargas útiles sin plantillas de asignación para las API de REST en API Gateway

6. Especifique el siguiente código en el editor de texto.

```
{
    "TableName":"data-transformation-tutorial-console-ddb",
    "Item": {
        "id": {
            "N": $input.json("$.id")
        },
        "type": {
            "S": $input.json("$.type")
        },
        "price": {
            "N": $input.json("$.price")
        }
    }
}
```

   Esta plantilla especifica la tabla como `data-transformation-tutorial-console-ddb` y establece los elementos como `id`, `type` y `price`. Los elementos procederán del cuerpo del método `POST`. También puede usar un modelo de datos para ayudar a crear una plantilla de mapeo. Para obtener más información, consulte Solicitud de la validación de las API de REST en API Gateway.

7. Elija Guardar para guardar la plantilla de mapeo.

Para añadir un método y una respuesta de integración desde el método **POST**

CloudFormation ha creado un método y una respuesta de integración en blanco. Editará esta respuesta para proporcionar más información. Para obtener más información sobre cómo editar las respuestas, consulte Ejemplos de asignación de parámetros para las API de REST en API Gateway.

1. En la pestaña Respuesta de integración, en Predeterminado: respuesta, seleccione Editar.
2. Elija Plantillas de mapeo y, a continuación, elija Agregar plantilla de mapeo.
3. En Content-type, introduzca **application/json**.
4. En el editor de código, introduzca la siguiente plantilla de mapeo de salida para enviar un mensaje de salida:

   ```
   { "message" : "Your response was recorded at $context.requestTime" }
   ```

   Para obtener más información acerca de las variables de contexto, consulte Variables de contexto para transformaciones de datos.
5. Elija Guardar para guardar la plantilla de mapeo.

Probar el método **POST**

Elija la pestaña Prueba. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña.

1. En el cuerpo de la solicitud, introduzca el siguiente ejemplo.

   ```
   {
             "id": "4",
             "type" : "dog",
             "price": "321"
   }
   ```
2. Seleccione Probar

   El resultado debería mostrar su mensaje de éxito.

   Puede abrir la consola de DynamoDB en https://console.aws.amazon.com/dynamodb/ para comprobar que el elemento de ejemplo esté en la tabla.

Para eliminar una pila de CloudFormation

1. Abra la consola de CloudFormation en https://console.aws.amazon.com/cloudformation.
2. Seleccione su pila de CloudFormation.
3. Elija Delete (Eliminar) y, a continuación, confirme su elección.

Configuración de la transformación de datos mediante la CLI de AWS

En este tutorial, creará una API y una tabla de DynamoDB incompletas con el siguiente archivo .zip data-transformation-tutorial-cli.zip. Esta API incompleta tiene un recurso `/pets` con un método `GET` integrado en el punto de conexión HTTP `http://petstore-demo-endpoint.execute-api.com/petstore/pets`. Creará un método `POST` para conectarse a una tabla de DynamoDB y utilizará plantillas de mapeo para introducir datos en una tabla de DynamoDB.

- Transformará los datos de salida según la plantilla de mapeo en the section called "Transformaciones de plantillas de asignación".
- Creará un método `POST` para permitir al usuario `POST` información de la mascota en una tabla de Amazon DynamoDB mediante una plantilla de mapeo.

Para crear una pila de CloudFormation

Descargue y descomprima la plantilla de creación de aplicaciones para CloudFormation.

Para completar el siguiente tutorial, necesita la versión 2 de la AWS Command Line Interface (AWS CLI).

Para comandos largos, se utiliza un carácter de escape (\) para dividir un comando en varias líneas.

> **Note**
>
> En Windows, algunos comandos de la CLI de Bash que utiliza habitualmente (por ejemplo, `zip`) no son compatibles con los terminales integrados del sistema operativo. Para obtener una versión de Ubuntu y Bash integrada con Windows, instale el subsistema de Windows para Linux. Los comandos de la CLI de ejemplo de esta guía utilizan el formato Linux. Los comandos que incluyen documentos JSON en línea deben reformatearse si utiliza la CLI de Windows.

1. Utilice el siguiente comando para crear la pila CloudFormation.

```
aws cloudformation create-stack --stack-name data-transformation-tutorial-cli
 --template-body file://data-transformation-tutorial-cli.zip --capabilities
 CAPABILITY_NAMED_IAM
```

2. CloudFormation aprovisiona los recursos especificados en la plantilla. Puede tardar varios minutos en finalizar el aprovisionamiento de los recursos. Use el siguiente comando para ver el estado de su pila CloudFormation.

```
aws cloudformation describe-stacks --stack-name data-transformation-tutorial-cli
```

3. Cuando el estado de su pila CloudFormation sea `StackStatus: "CREATE_COMPLETE"`, utilice el siguiente comando para recuperar los valores de salida relevantes para los pasos futuros.

```
aws cloudformation describe-stacks --stack-name data-transformation-tutorial-cli
--query "Stacks[*].Outputs[*].{OutputKey: OutputKey, OutputValue: OutputValue,
Description: Description}"
```

   A continuación se muestran los valores de salida:

   - ApiRole, que es el nombre del rol que permite a API Gateway incluir elementos en la tabla de DynamoDB. En este tutorial, el nombre de rol es `data-transformation-tutorial-cli-APIGatewayRole-ABCDEFG`.
   - DDBTableName, que es el nombre de la tabla de DynamoDB. En este tutorial, el nombre de la tabla es `data-transformation-tutorial-cli-ddb`.
   - ResourceId, que es el identificador del recurso de las mascotas donde están expuestos los métodos `GET` y `POST`. Para este tutorial, el ID del recurso es `efg456`.
   - ApiID, que es el identificador de la API. Para este tutorial, el ID de la API es `abc123`.

Para probar el método **`GET`** antes de la transformación de datos

- Utilice el siguiente comando para probar el método `GET`.

```
aws apigateway test-invoke-method --rest-api-id abc123 \
          --resource-id efg456 \
          --http-method GET
```

El resultado de la prueba mostrará lo siguiente.

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

Transformará este resultado según la plantilla de mapeo en the section called “Transformaciones de plantillas de asignación”.

Para transformar la respuesta de integración de **GET**

- Use el siguiente comando para actualizar la respuesta de integración del método GET. Reemplace *rest-api-id* y *resource-id* con sus valores.

  Utilice el siguiente comando para crear la respuesta de integración.

```
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]"}'
```

Para probar el método **GET**

- Utilice el siguiente comando para probar el método GET.

```
aws apigateway test-invoke-method --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method GET \
```

El resultado de la prueba mostrará la respuesta transformada.

```
[
  {
    "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
  }
]
```

Para crear un método **POST**

1. Use el siguiente comando para crear un método nuevo en el recurso /pets.

```
aws apigateway put-method --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method POST \
  --authorization-type "NONE" \
```

Este método le permitirá enviar información de mascotas a la tabla de DynamoDB que creó en la pila CloudFormation.

2. Utilice el siguiente comando para una integración Servicio de AWS en el método POST.

```
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. Utilice el siguiente comando para crear una respuesta de método para una llamada correcta del método POST.

```
aws apigateway put-method-response --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method POST \
  --status-code 200
```

4. Utilice el siguiente comando para crear una respuesta de integración para una llamada correcta del método POST.

```
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\"}"}'
```

Para probar el método **POST**

- Utilice el siguiente comando para probar el método POST.

```
aws apigateway test-invoke-method --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method POST \
  --body '{\"id\": \"4\", \"type\": \"dog\", \"price\": \"321\"}'
```

El resultado mostrará el mensaje correcto.

Para eliminar una pila de CloudFormation

- Use el siguiente comando para eliminar los recursos CloudFormation.

```
aws cloudformation delete-stack  --stack-name data-transformation-tutorial-cli
```

Plantilla CloudFormation de transformación de datos completada

El siguiente ejemplo es una plantilla CloudFormation completada, que crea una API y una tabla de DynamoDB con un recurso `/pets` con los métodos `GET` y `POST`.

- El método `GET` obtendrá datos del punto de conexión HTTP `http://petstore-demo-endpoint.execute-api.com/petstore/pets`. Los datos de salida se transformarán según la plantilla de mapeo en the section called “Transformaciones de plantillas de asignación”.
- El método `POST` permitirá al usuario `POST` información de la mascota en una tabla de DynamoDB mediante una plantilla de mapeo.

Plantilla de CloudFormation de ejemplo

```
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
```

Ejemplos que utilizan variables para las transformaciones de plantilla de asignación para API Gateway

En los siguientes ejemplos se muestra cómo utilizar las variables `$context`, `input` y `util` en las plantillas de asignación. Puede utilizar una integración simulada o una integración sin proxy de Lambda que devuelva el evento de entrada a API Gateway. Para obtener una lista de todas las variables admitidas para las transformaciones de datos, consulte the section called “Variables para transformaciones de datos”.

Ejemplo 1: pase de múltiples variables **`$context`** al punto de conexión de integración

El siguiente ejemplo muestra una plantilla de asignación que mapea variables `$context` de entrada a las variables de backend con nombres ligeramente diferentes en una carga de la solicitud de integración:

```
{
    "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"
}
```

La salida de esta plantilla de asignación debe tener el siguiente aspecto:

```
{
  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'
}
```

Una de las variables es una clave de API. En este ejemplo se supone que el método requiere una clave de API.

Ejemplo 2: pase de todos los parámetros de solicitud al punto de conexión de integración a través de una carga útil JSON

El siguiente ejemplo pasa todos los parámetros de solicitud, como los parámetros `path`, `querystring` y `header`, a través del punto de conexión de integración mediante una carga útil de JSON:

```
#set($allParams = $input.params())
{
  "params" : {
    #foreach($type in $allParams.keySet())
    #set($params = $allParams.get($type))
    "$type" : {
      #foreach($paramName in $params.keySet())
      "$paramName" : "$util.escapeJavaScript($params.get($paramName))"
      #if($foreach.hasNext),#end
```

```
      #end
    }
    #if($foreach.hasNext),#end
    #end
  }
}
```

Si una solicitud tiene los siguientes parámetros de entrada:

- Un parámetro de ruta denominado `myparam`
- Parámetros de cadenas de consulta `querystring1=value1,value2`
- Encabezados `"header1" : "value1"`.

La salida de esta plantilla de asignación debe tener el siguiente aspecto:

```
{"params":{"path":{"example2":"myparamm"},"querystring":
{"querystring1":"value1,value2"},"header":{"header1":"value1"}}}
```

Ejemplo 3: pase de una subsección de una solicitud de método al punto de conexión de integración

En el siguiente ejemplo se utiliza el parámetro de entrada `name` para recuperar solo el parámetro `name` y el parámetro de entrada `input.json('$')` para recuperar todo el cuerpo de la solicitud de método:

```
{
    "name" : "$input.params('name')",
    "body" : $input.json('$')
}
```

Para una solicitud que incluye los parámetros de cadena de consulta `name=Bella&type=dog` y el cuerpo siguiente:

```
{
    "Price" : "249.99",
    "Age": "6"
}
```

La salida de esta plantilla de asignación debe tener el siguiente aspecto:

```
{
```

```
    "name" : "Bella",
    "body" : {"Price":"249.99","Age":"6"}
}
```

Esta plantilla de asignación elimina el parámetro de cadena de consulta `type=dog`.

Si la entrada JSON contiene caracteres sin escape que no puede analizar JavaScript, es posible que API Gateway devuelva una respuesta 400. Aplique `$util.escapeJavaScript($input.json('$'))` para asegurarse de que la entrada JSON se pueda analizar correctamente.

El ejemplo anterior con `$util.escapeJavaScript($input.json('$'))` se aplica de la siguiente manera:

```
{
    "name" : "$input.params('name')",
    "body" : "$util.escapeJavaScript($input.json('$'))"
}
```

En este caso, la salida de esta plantilla de asignación debe tener el siguiente aspecto:

```
{
    "name" : "Bella",
    "body": {"Price":"249.99","Age":"6"}
}
```

Ejemplo 4: uso de una expresión JSONPath para pasar una subsección de una solicitud de método al punto de conexión de integración

En el siguiente ejemplo se utilizan expresiones JSONPath para recuperar solo el parámetro de entrada `name` y el valor de `Age` del cuerpo de solicitud:

```
{
    "name" : "$input.params('name')",
    "body" : $input.json('$.Age')
}
```

Para una solicitud que incluye los parámetros de cadena de consulta `name=Bella&type=dog` y el cuerpo siguiente:

```
{
```

```
    "Price" : "249.99",
    "Age": "6"
}
```

La salida de esta plantilla de asignación debe tener el siguiente aspecto:

```
{
    "name" : "Bella",
    "body" : "6"
}
```

Esta plantilla de asignación elimina el parámetro de cadena de consulta `type=dog` y el campo `Price` del cuerpo.

Si una carga de solicitud de método contiene caracteres sin escape que no puede analizar JavaScript, es posible que API Gateway devuelva una respuesta `400`. Aplique `$util.escapeJavaScript()` para asegurarse de que la entrada JSON se pueda analizar correctamente.

El ejemplo anterior con `$util.escapeJavaScript($input.json('$.Age'))` se aplica de la siguiente manera:

```
{
    "name" : "$input.params('name')",
    "body" : "$util.escapeJavaScript($input.json('$.Age'))"
}
```

En este caso, la salida de esta plantilla de asignación debe tener el siguiente aspecto:

```
{
    "name" : "Bella",
    "body": "\"6\""
}
```

Ejemplo 5: uso de una expresión JSONPath para pasar información sobre una solicitud de método al punto de conexión de integración

En el siguiente ejemplo se utilizan `$input.params()`, `$input.path()` y `$input.json()` para enviar información sobre una solicitud de método al punto de conexión de integración. Esta plantilla de asignación utiliza el método `size()` para proporcionar el número de elementos de una lista.

```
{
    "id" : "$input.params('id')",
    "count" : "$input.path('$.things').size()",
    "things" : $input.json('$.things')
}
```

Para una solicitud que incluye el parámetros de ruta `123` y el cuerpo siguiente:

```
{
      "things": {
            "1": {},
            "2": {},
            "3": {}
      }
}
```

La salida de esta plantilla de asignación debe tener el siguiente aspecto:

```
{"id":"123","count":"3","things":{"1":{},"2":{},"3":{}}}
```

Si una carga de solicitud de método contiene caracteres sin escape que no puede analizar JavaScript, es posible que API Gateway devuelva una respuesta `400`. Aplique `$util.escapeJavaScript()` para asegurarse de que la entrada JSON se pueda analizar correctamente.

El ejemplo anterior con `$util.escapeJavaScript($input.json('$.things'))` se aplica de la siguiente manera:

```
{
     "id" : "$input.params('id')",
     "count" : "$input.path('$.things').size()",
     "things" : "$util.escapeJavaScript($input.json('$.things'))"
}
```

La salida de esta plantilla de asignación debe tener el siguiente aspecto:

```
{"id":"123","count":"3","things":"{\"1\":{},\"2\":{},\"3\":{}}"}
```

Variables para transformaciones de datos para API Gateway

Cuando cree una asignación de parámetro, puede utilizar variables de contexto como origen de datos. Cuando cree las transformaciones de plantilla de asignación, puede utilizar variables de contexto, de entrada y de utilidad en los scripts que escriba en Velocity Template Language (VTL). Por obtener ejemplos de plantillas de asignación que utilizan estas variables de referencia, consulte the section called “Ejemplos que utilizan variables para las transformaciones de plantilla de asignación”.

Para obtener una lista de variables de referencia para el registro de acceso, consulte the section called “Variables para el registro de acceso para API Gateway”.

Variables de contexto para transformaciones de datos

Puede utilizar las siguientes variables `$context` que distinguen entre mayúsculas y minúsculas para transformaciones de datos.

Parámetro	Descripción
`$context.accountId`	El ID de cuenta de AWS del propietario de la API.
`$context.apiId`	El identificador que API Gateway asigna a su API.
`$context.authorizer.claims.`*`property`*	Una propiedad de las notificaciones devueltas por el grupo de usuarios de Amazon Cognito una vez que se ha autenticado correctamente al intermediario del método. Para obtener más información, consulte the section called “Uso de grupos de usuarios de Amazon Cognito como autorizador para una API REST”. **Note** La llamada a `$context.authorizer.claims` devuelve null.

Parámetro	Descripción
`$context.authorizer.principalId`	La identificación de usuario principal asociada con el token enviado por el cliente y devuelto por el autorizador de Lambda de API Gateway (que anteriormente se denominaba autorizador personalizado). Para obtener más información, consulte the section called “Uso de autorizadores Lambda”.

`$context.authorizer.` *`property`*	El valor en forma de cadena del par clave-valor especificado de la asignación `context` que devuelve la función de Lambda del autorizador de Lambda de API Gateway. Por ejemplo, si el autorizador devuelve la siguiente asignación de `context`:
	
	`"context" : {`
	`  "key": "value",`
	`  "numKey": 1,`
	`  "boolKey": true`
	`}`
	
	La llamada a `$context.authorizer.key` devuelve la cadena `"value"`, la llamada a `$context.authorizer.numKey` devuelve la cadena `"1"` y la llamada a `$context.authorizer.boolKey` devuelve la cadena `"true"`.
	
	En el caso de la *`propiedad`* , el único carácter especial admitido es el carácter `(_)` de subrayado.
	
	Para obtener más información, consulte the section called “Uso de autorizadores Lambda”.

`$context.awsEndpointRequestId`	El ID de solicitud del punto de enlace de AWS.

Parámetro	Descripción
`$context.deploymentId`	El ID de la implementación de la API.
`$context.domainName`	El nombre de dominio completo que se utiliza para invocar la API. Este debe ser el mismo que el encabezado `Host` entrante.
`$context.domainPrefix`	La primera etiqueta del `$context.domainName` .
`$context.error.message`	Una cadena que contiene un mensaje de error de API Gateway. Esta variable solo se puede usar para una sustitución sencilla de variables en una plantilla de mapeo de cuerpo GatewayResponse, que no procesa el motor de Velocity Template Language, y en el registro de acceso. Para obtener más información, consulte the section called “Métricas” y the section called “Configuración de respuestas de gateway para personalizar respuestas de errores”.
`$context.error.messageString`	El valor entrecomillado de `$context.error.message` , es decir, `"$context.error.message"` .
`$context.error.responseType`	Un tipo de GatewayResponse. Esta variable solo se puede usar para una sustitución sencilla de variables en una plantilla de mapeo de cuerpo GatewayResponse, que no procesa el motor de Velocity Template Language, y en el registro de acceso. Para obtener más información, consulte the section called “Métricas” y the section called “Configuración de respuestas de gateway para personalizar respuestas de errores”.

Parámetro	Descripción
`$context.error.validationErrorString`	Una cadena que contiene un mensaje de error de validación detallado.
`$context.extendedRequestId`	El ID ampliado que API Gateway genera y asigna a la solicitud de API. El ID de solicitud ampliado contiene información útil para la depuración y la resolución de problemas.
`$context.httpMethod`	El método HTTP utilizado. Los valores válidos son: `DELETE`, `GET`, `HEAD`, `OPTIONS`, `PATCH`, `POST` y `PUT`.
`$context.identity.accountId`	El ID de cuenta de AWS asociado con la solicitud.
`$context.identity.apiKey`	Para los métodos de API que necesitan una clave de API, esta variable es la clave de API asociada a la solicitud del método. Para métodos que no requieren una clave de API, esta variable corresponde a valores null. Para obtener más información, consulte the section called “Planes de uso”.
`$context.identity.apiKeyId`	El ID de clave de API asociado a una solicitud de API que requiere una clave de API.
`$context.identity.caller`	El identificador principal del intermediario que firmó la solicitud. Compatible con recursos que utilizan la autorización de IAM.

Parámetro	Descripción
`$context.identity.cognitoAuthenticationProvider`	Una lista separada por comas de todos los proveedores de autenticación de Amazon Cognito utilizados por el intermediario que realiza la solicitud. Solo está disponible si la solicitud se firmó con las credenciales de Amazon Cognito. Por ejemplo, para una identidad de un grupo de usuarios de Amazon Cognito, `cognito-idp.`*`region`*`.amazonaws.com/`*`user_pool_id`* `,cognito-idp.`*`region`*`.amazonaws.com/`*`user_pool_id`* `:CognitoSignIn:`*`token subject claim`* Consulte Uso de las identidades federadas en la Guía para desarrolladores de Amazon Cognito para obtener información sobre los proveedores de autenticación de Amazon Cognito disponibles.
`$context.identity.cognitoAuthenticationType`	El tipo de autenticación de Amazon Cognito del intermediario que realiza la solicitud. Solo está disponible si la solicitud se firmó con las credenciales de Amazon Cognito. Los valores posibles incluyen `authenticated` para identidades autenticadas y `unauthenticated` para identidades no autenticadas.
`$context.identity.cognitoIdentityId`	El ID de identidad de Amazon Cognito del intermediario que realiza la solicitud. Solo está disponible si la solicitud se firmó con las credenciales de Amazon Cognito.

Parámetro	Descripción
`$context.identity.cognitoId entityPoolId`	El ID del grupo de identidades de Amazon Cognito del intermediario que realiza la solicitud. Solo está disponible si la solicitud se firmó con las credenciales de Amazon Cognito.
`$context.identity.principalOrgId`	El ID de organización de AWS.
`$context.identity.sourceIp`	La dirección IP de origen de la conexión TCP inmediata que realiza la solicitud al punto de conexión de API Gateway.
`$context.identity.clientCer t.clientCertPem`	El certificado de cliente codificado en PEM que el cliente presentó durante la autentica ción TLS mutua. Presente cuando un cliente accede a una API mediante un nombre de dominio personalizado que tiene una TLS mutua habilitada. Presente solo en los registros de acceso si falla la autenticación TLS mutua.
`$context.identity.clientCer t.subjectDN`	Nombre distintivo del asunto del certificado que presenta un cliente. Presente cuando un cliente accede a una API mediante un nombre de dominio personalizado que tiene una TLS mutua habilitada. Presente solo en los registros de acceso si falla la autenticación TLS mutua.
`$context.identity.clientCer t.issuerDN`	Nombre distintivo del emisor del certificado que presenta un cliente. Presente cuando un cliente accede a una API mediante un nombre de dominio personalizado que tiene una TLS mutua habilitada. Presente solo en los registros de acceso si falla la autenticación TLS mutua.

Parámetro	Descripción
`$context.identity.clientCert.serialNumber`	Número de serie del certificado. Presente cuando un cliente accede a una API mediante un nombre de dominio personalizado que tiene una TLS mutua habilitada. Presente solo en los registros de acceso si falla la autenticación TLS mutua.
`$context.identity.clientCert.validity.notBefore`	Fecha antes de la cual el certificado no es válido. Presente cuando un cliente accede a una API mediante un nombre de dominio personalizado que tiene una TLS mutua habilitada. Presente solo en los registros de acceso si falla la autenticación TLS mutua.
`$context.identity.clientCert.validity.notAfter`	Fecha después de la cual el certificado no es válido. Presente cuando un cliente accede a una API mediante un nombre de dominio personalizado que tiene una TLS mutua habilitada. Presente solo en los registros de acceso si falla la autenticación TLS mutua.
`$context.identity.vpcId`	El ID de VPC de la VPC que realiza la solicitud al punto de conexión de API Gateway.
`$context.identity.vpceId`	El ID de punto de conexión de VPC del punto de conexión de VPC que realiza la solicitud al punto de conexión de API Gateway. Está presente solo cuando tiene una API privada.
`$context.identity.user`	El identificador principal del usuario que se autorizará a acceder al recurso. Compatible con recursos que utilizan la autorización de IAM.
`$context.identity.userAgent`	El encabezado `User-Agent` del intermediario de la API.

Parámetro	Descripción
`$context.identity.userArn`	El Nombre de recurso de Amazon (ARN) del usuario identificado después de la autentica ción. Para obtener más información, consulte https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html.
`$context.isCanaryRequest`	Devuelve `true` si la solicitud se dirigió al canario y `false` si no se dirigió al canario. Está presente solo cuando tiene un canario activado.
`$context.path`	Ruta de acceso de la solicitud. Por ejemplo, en el caso del URL de una solicitud que no es de proxy `https://{rest-api-id}.execute-api.{region}.amazonaws.com/{stage}/root/child` , el valor de `$context.path` es `/{stage}/root/child` .
`$context.protocol`	Protocolo de la solicitud; por ejemplo, `HTTP/1.1`. **Note** Las API de API Gateway pueden aceptar solicitudes HTTP/2, pero API Gateway envía solicitudes a las integraciones de backend mediante HTTP/1.1. Como resultado, el protocolo de solicitud se registra como HTTP/1.1 incluso si un cliente envía una solicitud que usa HTTP/2.

Parámetro	Descripción
`$context.requestId`	ID de la solicitud. Los clientes pueden anular este ID de solicitud. Utilice `$context.extendedRequestId` para un ID de solicitud único que genera API Gateway.
`$context.requestOverride.header.` *`header_name`*	La invalidación del encabezado de solicitud. Si este parámetro se define, contiene los encabezados que se utilizarán en lugar de los HTTP Headers (Encabezados HTTP) que se definen en el panel Integration Request (Solicitud de integración). Para obtener más información, consulte Anulación de los parámetros de solicitud y respuesta y de los códigos de estado de la API para las API de REST en API Gateway.
`$context.requestOverride.path.` *`path_name`*	La invalidación de la ruta de acceso de la solicitud. Si este parámetro se define, contiene la ruta de acceso de la solicitud que se utilizará en lugar de los URL Path Parameters (Parámetros de ruta URL) que se definen en el panel Integration Request (Solicitud de integración). Para obtener más información, consulte Anulación de los parámetros de solicitud y respuesta y de los códigos de estado de la API para las API de REST en API Gateway.

Parámetro	Descripción
`$context.requestOverride.querystring.` *querystring_name*	La invalidación de la cadena de consulta de solicitud. Si este parámetro se define, contiene las cadenas de consulta de solicitud que se utilizarán en lugar de los URL Query String Parameters (Parámetros de cadena de consulta URL) que se definen en el panel Integration Request (Solicitud de integración). Para obtener más información, consulte Anulación de los parámetros de solicitud y respuesta y de los códigos de estado de la API para las API de REST en API Gateway.
`$context.responseOverride.header.` *header_name*	La invalidación del encabezado de respuesta. Si este parámetro se define, contiene el encabezado que se devolverá en lugar del Response header (Encabezado de respuesta) que se define como Default mapping (Asignación predeterminada) en el panel Integration Response (Respuesta de integración). Para obtener más información, consulte Anulación de los parámetros de solicitud y respuesta y de los códigos de estado de la API para las API de REST en API Gateway.

Parámetro	Descripción
`$context.responseOverride.status`	La invalidación del código de estado de respuesta. Si este parámetro se define, contiene el código de estado que se devolverá en lugar del Method response status (Estado de respuesta de método) que se define como Default mapping (Asignación predeterminada) en el panel Integration Response (Respuesta de integración). Para obtener más información, consulte Anulación de los parámetros de solicitud y respuesta y de los códigos de estado de la API para las API de REST en API Gateway.
`$context.requestTime`	Hora de la solicitud en formato CLF-(`dd/MMM/yyyy:HH:mm:ss +-hhmm`).
`$context.requestTimeEpoch`	Hora de la solicitud en formato Epoch en milisegundos.
`$context.resourceId`	El identificador que API Gateway asigna a su recurso.
`$context.resourcePath`	La ruta a su recurso. Por ejemplo, en el caso del URI de una solicitud que no es de proxy `https://{rest-api-id}.execute-api.{region}.amazonaws.com/{stage}/root/child`, el valor de `$context.resourcePath` es `/root/child`. Para obtener más información, consulte Tutorial: Creación de una API de REST con integración no de proxy HTTP.
`$context.stage`	La etapa de implementación de la solicitud de la API (por ejemplo, `Beta` o `Prod`).

Parámetro	Descripción
`$context.wafResponseCode`	La respuesta recibida desde AWS WAF: `WAF_ALLOW` o `WAF_BLOCK`. No se establecerá si la etapa no está asociada a una ACL web. Para obtener más información, consulte the section called “AWS WAF”.
`$context.webaclArn`	El ARN completo de la ACL web que se utiliza para decidir si se debe permitir o bloquear la solicitud. No se establecerá si la etapa no está asociada a una ACL web. Para obtener más información, consulte the section called “AWS WAF”.

Variables de entrada

Puede utilizar las siguientes variables `$input` que distinguen entre mayúsculas y minúsculas para hacer referencia a la carga útil de solicitud del método y a los parámetros de solicitud del método. Están disponibles las siguientes funciones:

Variable y función	Descripción
`$input.body`	Devuelve la carga de solicitud bruta como una cadena. Puede utilizar `$input.body` para conservar números enteros de punto flotante, como `10.00`.
`$input.json(x)`	Esta función evalúa una expresión JSONPath y devuelve los resultados como una cadena JSON. Por ejemplo, `$input.json('$.pets')` devuelve una cadena JSON que representa la estructura de `pets` (mascotas).

Variable y función	Descripción
	Para obtener más información acerca de JSONPath, consulte JSONPath o JSONPath for Java.
`$input.params()`	Devuelve una asignación de todos los parámetros de solicitud. Recomendamos que se utilice `$util.escapeJavaScript` para sanear el resultado a fin de evitar un posible ataque de inyección. Para un control total del saneamiento de solicitudes, utilice una integración de proxy sin plantilla y gestione dicho saneamiento en su integración.
`$input.params(x)`	Devuelve el valor de un parámetro de solicitud de método de la ruta, cadena de consulta o valor de encabezado (buscado en este orden) dada una cadena de nombre de parámetro `x`. Recomendamos que se utilice `$util.escapeJavaScript` para sanear el parámetro a fin de evitar un posible ataque de inyección. Para un control total del saneamiento de parámetros, utilice una integración de proxy sin plantilla y gestione dicho saneamiento en su integración.

Variable y función	Descripción
`$input.path(x)`	Toma una cadena de expresión JSONPath (`x`) y devuelve una representación del resultado en forma de objeto JSON. Esto le permite tener acceso y manipular los elementos de la carga de forma nativa en Apache Velocity Template Language (VTL). Por ejemplo, si la expresión `$input.path('$.pets')` devuelve un objeto como este: `[ { "id": 1, "type": "dog", "price": 249.99 }, { "id": 2, "type": "cat", "price": 124.99 }, { "id": 3, "type": "fish", "price": 0.99 } ]` `$input.path('$.pets').size()` devolvería `"3"`. Para obtener más información acerca de JSONPath, consulte JSONPath o JSONPath for Java.

Variables de etapa

Puede utilizar las siguientes variables de etapa como marcadores de posición para ARN y URL en integraciones de método. Para obtener más información, consulte the section called “Uso de variables de etapa”.

Sintaxis	Descripción
`$stageVariables.` *`variable_name`* , `$stageVariables['` *`variable_name`* `']`, o bien `${stageVariables['` *`variable_name`* `']}`	*`variable_name`* representa un nombre de variable de etapa.

Variables de utilidad

Puede utilizar las siguientes variables `$util` que distinguen entre mayúsculas y minúsculas para utilizar funciones de utilidad en las plantillas de asignación. A menos que se especifique lo contrario, el conjunto de caracteres predeterminado es UTF-8.

Función	Descripción
`$util.escapeJavaScript()`	Aplica caracteres de escape a los caracteres de una cadena usando reglas de cadena de JavaScript. **Note** Esta función convertirá todas las comillas simples (`'`) en caracteres de escape (`\'`). Sin embargo, JSON no admite comillas simples con caracteres de escape. Por lo tanto, cuando la salida de esta función se utiliza en una propiedad de JSON, debe convertir todas las comillas simples con caracteres de escape (`\'`) en comillas

Función	Descripción
	simples normales (`'`). Esto se muestra en el siguiente ejemplo: `"input" : "$util.escapeJavaScript( data).replaceAll("\\'",""')"`

`$util.parseJson()`	Toma un elemento JSON en forma de cadena y devuelve una representación del resultado en forma de objeto. Puede utilizar el resultado de esta función para tener acceso y manipular los elementos de la carga de forma nativa en Apache Velocity Template Language (VTL). Por ejemplo, si tiene la siguiente carga:
	`{"errorMessage":"{\"key1\":\"var1\", \"key2\":{\"arr\":[1,2,3]}}"}`
	y usa la siguiente plantilla de asignación
	`#set ($errorMessageObj = $util.parseJson($input.path('$.errorMessage')))`
	`{`
	`"errorMessageObjKey2ArrVal" : $errorMessageObj.key2.arr[0]`
	`}`
	Obtendrá el siguiente resultado:
	`{`
	`"errorMessageObjKey2ArrVal" : 1`
	`}`

`$util.urlEncode()`	Convierte una cadena en formato "application/x-www-form-urlencoded".

Función	Descripción
`$util.urlDecode()`	Descodifica una cadena "application/x-www-form-urlencoded".
`$util.base64Encode()`	Codifica los datos en una cadena codificada en base64.
`$util.base64Decode()`	Descodifica los datos de una cadena codificada en base64.

Respuestas de Gateway para las API de REST en API Gateway

Una respuesta de gateway se identifica mediante un tipo de respuesta definido por API Gateway. La respuesta se compone de un código de estado HTTP, un conjunto de encabezados adicionales especificados mediante asignaciones de parámetros y una carga útil generada por una plantilla de asignación distinta de VTL.

En la API REST de API Gateway, una respuesta de gateway se representa mediante GatewayResponse. En OpenAPI, una instancia de `GatewayResponse` se describe mediante la extensión x-amazon-apigateway-gateway-responses.gatewayResponse.

Para habilitar una respuesta de gateway, debe configurar una para un tipo de respuesta admitido en el nivel de API. Siempre que API Gateway devuelve una respuesta de este tipo, se aplican las plantillas de mapeo de encabezados y cargas definidas en la respuesta de gateway para devolver los resultados asignados al intermediario de la API.

En la siguiente sección, le mostramos cómo configurar respuestas de gateway utilizando la consola de API Gateway y la API REST de API Gateway.

Configuración de respuestas de gateway para personalizar respuestas de errores

Si API Gateway no puede procesar una solicitud entrante, devuelve al cliente una respuesta de error sin reenviar la solicitud al backend de integración. De forma predeterminada, la respuesta de error contiene un breve mensaje de error descriptivo. Por ejemplo, si intenta llamar a una operación en un recurso de API no definido, recibe una respuesta de error con el mensaje `{ "message": "Missing Authentication Token" }`. Si es la primera vez que utiliza API Gateway, es posible que le resulte difícil entender qué es lo que ha ocurrido.

Para algunas de las respuestas de error, API Gateway permite a los desarrolladores de API que personalicen las respuestas de forma que devuelvan las respuestas en formatos diferentes. Para el ejemplo de `Missing Authentication Token`, puede añadir una pista a la carga de respuesta original con la posible causa, como en este ejemplo: `{"message":"Missing Authentication Token", "hint":"The HTTP method or resources may not be supported."}`.

Cuando la API es un mediador entre un intercambio externo y AWS Cloud, se utilizan plantillas de asignación de VTL para la solicitud de integración o la respuesta de integración para asignar la carga de un formato a otro. Sin embargo, las plantillas de asignación de VTL solo funcionan para solicitudes válidas con respuestas correctas.

En el caso de las solicitudes no válidas, API Gateway omite la integración en su totalidad y devuelve una respuesta de error. Debe utilizar la personalización para representar las respuestas de error en un formato compatible con el intercambio. Aquí, la personalización se presenta en una plantilla de asignación que no es de VTL que solo admite sustituciones de variables sencillas.

Con el fin de generalizar la respuesta de error generada por API Gateway a cualquier respuesta generada por API Gateway, nos referimos a estas respuestas como respuesta de gateway. Esto permite distinguir las respuestas generadas por API Gateway de las respuestas de integración. Una plantilla de asignación de respuesta de gateway puede tener acceso a los valores de la variable `$context` y a los valores de la propiedad `$stageVariables`, así como a los parámetros de solicitud de método, con el formato `method.request.`*`param-position`*`.`*`param-name`*.

Para obtener más información acerca de las variables `$context`, consulte Variables de contexto para transformaciones de datos. Para obtener más información acerca de `$stageVariables`, consulte Variables de etapa. Para obtener más información sobre los parámetros de solicitud de método, consulte the section called “Variables de entrada”.

Temas

- Configurar una respuesta de gateway para una API REST mediante la consola de API Gateway
- Configurar una respuesta de gateway mediante la API REST de API Gateway
- Configurar la personalización de respuestas de gateway en OpenAPI
- Tipos de respuestas de puerta de enlace para API Gateway

Configurar una respuesta de gateway para una API REST mediante la consola de API Gateway

En el siguiente ejemplo se muestra cómo configurar una respuesta de puerta de enlace para una API de REST mediante la consola de API Gateway

Para personalizar una respuesta de gateway mediante la consola de API Gateway

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST.
3. En el panel de navegación principal, elija Respuestas de puerta de enlace.
4. Elija un tipo de respuesta y, a continuación, elija Editar. En este tutorial, utilizamos Falta el token de autenticación como ejemplo.
5. Puede cambiar el Código de estado generado por API Gateway para devolver un código de estado diferente según los requisitos de la API. En este ejemplo, la personalización cambia el código de estado del valor predeterminado (`403`) a `404`, porque este mensaje de error se produce cuando un cliente llama a un recurso no admitido o no válido que se considera no encontrado.
6. Para devolver encabezados personalizados, elija Agregar encabezado de respuesta en Encabezados de respuesta. Con fines ilustrativos, añadimos los siguientes encabezados personalizados:

   ```
   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
   ```

   En las asignaciones de encabezado anteriores, un nombre de dominio estático (`'a.b.c'`) se asigna al encabezado `Allow-Control-Allow-Origin` para permitir el acceso de CORS a la API; el encabezado de solicitud de entrada `x-amzn-RequestId` se asigna a `request-id` en la respuesta, la variable de ruta `petId` de la solicitud entrante se asigna al encabezado `request-path` de la respuesta y el parámetro de consulta q de la solicitud original se asigna al encabezado `request-query` de la respuesta.
7. En Plantillas de respuesta, mantenga `application/json` para Tipo de contenido e ingrese la siguiente plantilla de mapeo del cuerpo en el editor Cuerpo de la plantilla:

   ```
   {
   ```

```
    "message":"$context.error.messageString",
    "type": "$context.error.responseType",
    "statusCode": "'404'",
    "stage": "$context.stage",
    "resourcePath": "$context.resourcePath",
    "stageVariables.a": "$stageVariables.a"
}
```

Este ejemplo muestra cómo asignar las propiedades `$context` y `$stageVariables` a propiedades del cuerpo de la respuesta de gateway.

8. Seleccione Save changes (Guardar cambios).
9. Implemente la API en una etapa nueva o existente.

Pruebe la respuesta de la puerta de enlace llamando al siguiente comando CURL si la URL de invocación del método de API correspondiente es `https://o81lxisefl.execute-api.us-east-1.amazonaws.com/custErr/pets/{petId}`:

```
curl -v -H 'x-amzn-RequestId:123344566' https://o81lxisefl.execute-api.us-
east-1.amazonaws.com/custErr/pets/5/type?q=1
```

Como el parámetro de cadena de consulta adicional q=1 no es compatible con la API, se devuelve un error de la respuesta de puerta de enlace especificada. Debería obtener una respuesta de gateway similar a la siguiente:

```
> 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
}
```

En el ejemplo anterior se presupone que el backend de la API es Pet Store y que la API tiene un variable de etapa, a, definida.

Configurar una respuesta de gateway mediante la API REST de API Gateway

Antes de personalizar un respuesta de gateway mediante la API REST de API Gateway, debe haber creado una API y haber obtenido su identificador. Para recuperar el identificador de la API, puede seguir la relación de enlace restapi:gateway-responses y examinar el resultado.

Para personalizar una respuesta de gateway mediante la API REST de API Gateway

1. Para sobrescribir una instancia completa de GatewayResponse llame a la acción gatewayresponse:put. Especifique un responseType en el parámetro de ruta de URL y proporcione en la carga útil de la solicitud los mapeos statusCode, responseParameters y responseTemplates.
2. Para actualizar parte de una instancia de `GatewayResponse`, llame a la acción gatewayresponse:update. Especifique el parámetro `responseType` deseado en la ruta URL y proporcione en la carga útil de la solicitud las propiedades individuales `GatewayResponse` que desee, por ejemplo, el mapeo `responseParameters` o `responseTemplates`.

Configurar la personalización de respuestas de gateway en OpenAPI

Puede utilizar la extensión `x-amazon-apigateway-gateway-responses` en el nivel de raíz de la API para personalizar respuestas de gateway en OpenAPI. La siguiente definición de OpenAPI muestra un ejemplo de personalización de GatewayResponse del tipo `MISSING_AUTHENTICATION_TOKEN`.

```
  "x-amazon-apigateway-gateway-responses": {
    "MISSING_AUTHENTICATION_TOKEN": {
      "statusCode": 404,
      "responseParameters": {
        "gatewayresponse.header.x-request-path": "method.input.params.petId",
        "gatewayresponse.header.x-request-query": "method.input.params.q",
        "gatewayresponse.header.Access-Control-Allow-Origin": "'a.b.c'",
        "gatewayresponse.header.x-request-header": "method.input.params.Accept"
      },
      "responseTemplates": {
        "application/json": "{\n     \"message\": $context.error.messageString,\n
    \"type\":  \"$context.error.responseType\",\n     \"stage\":  \"$context.stage
\",\n     \"resourcePath\":  \"$context.resourcePath\",\n     \"stageVariables.a\":
 \"$stageVariables.a\",\n     \"statusCode\": \"'404'\"\n}"
      }
    }
```

En este ejemplo, la personalización cambia el código de estado del valor predeterminado (`403`) a `404`. También añade a la respuesta de gateway cuatro parámetros de encabezado y una plantilla de asignación de cuerpo para el tipo de medio `application/json`.

Tipos de respuestas de puerta de enlace para API Gateway

API Gateway expone las siguientes respuestas de gateway para su personalización por desarrolladores de la API.

Tipo de respuesta de gateway	Código de estado predeterminado	Descripción
`ACCESS_DENIED`	`403`	La respuesta de gateway para un error de autorización (por ejemplo, cuando un autorizador de Amazon Cognito o personalizado deniega el acceso). Si no se especifica el tipo de respuesta, se asigna el tipo `DEFAULT_4XX` .
`API_CONFIGURATION_ERROR`	`500`	La respuesta de gateway para una configuración de

Tipo de respuesta de gateway	Código de estado predeterminado	Descripción
		API no válida, incluida cuando una dirección de punto de enlace no válida enviada, cuando falla la descodificación base64 en datos binarios, cuando no se ha habilitado la compatibilidad con los datos binarios o cuando un mapeo de respuesta de integración que no coincide con ninguna plantilla y no se ha configurado ninguna plantilla predeterminada. Si no se especifica el tipo de respuesta, se asigna el tipo `DEFAULT_5XX` .
`AUTHORIZER_CONFIGURATION_ERROR`	`500`	La respuesta de gateway por no poder conectarse a un autorizador de Amazon Cognito o personalizado. Si no se especifica el tipo de respuesta, se asigna el tipo `DEFAULT_5XX` .
`AUTHORIZER_FAILURE`	`500`	La respuesta de gateway cuando un autorizador de Amazon Cognito o personalizado no puede autenticar al intermediario. Si no se especifica el tipo de respuesta, se asigna el tipo `DEFAULT_5XX` .

Tipo de respuesta de gateway	Código de estado predeterminado	Descripción
BAD_REQUEST_PARAMETERS	400	La respuesta de gateway cuando el parámetro de la solicitud no se puede validar de acuerdo con un validador de solicitudes habilitado. Si no se especifica el tipo de respuesta, se asigna el tipo DEFAULT_4XX .
BAD_REQUEST_BODY	400	La respuesta de gateway cuando el cuerpo de la solicitud no se puede validar de acuerdo con un validador de solicitudes habilitado. Si no se especifica el tipo de respuesta, se asigna el tipo DEFAULT_4XX .

Tipo de respuesta de gateway	Código de estado predeterminado	Descripción
DEFAULT_4XX	Null	La respuesta de gateway predeterminada para un tipo de respuesta sin especificar con el código de estado 4XX. Si se cambia el código de estado de esta respuesta de gateway, se cambian los códigos de estado de todas las demás respuestas 4XX al nuevo valor. Si se restablece este código de estado a null, se revierten los códigos de estado de todas las demás respuestas 4XX a sus valores originales. **Note** Las respuestas personalizadas de AWS WAF tienen prioridad sobre las respuestas de gateway personalizadas.

Tipo de respuesta de gateway	Código de estado predeterminado	Descripción
DEFAULT_5XX	Null	La respuesta de gateway predeterminada para un tipo de respuesta sin especificar con un código de estado 5XX. Si se cambia el código de estado de esta respuesta de gateway, se cambian los códigos de estado de todas las demás respuestas 5XX al nuevo valor. Si se restablece este código de estado a null, se revierten los códigos de estado de todas las demás respuestas 5XX a sus valores originales.
EXPIRED_TOKEN	403	La respuesta de gateway para un error de token de autenticación de AWS caducado. Si no se especifica el tipo de respuesta, se asigna el tipo DEFAULT_4XX .
INTEGRATION_FAILURE	504	La respuesta de gateway para un error de integración. Si no se especifica el tipo de respuesta, se asigna el tipo DEFAULT_5XX .

Tipo de respuesta de gateway	Código de estado predeterminado	Descripción
INTEGRATION_TIMEOUT	504	La respuesta de gateway para un error de tiempo de espera de la integración agotado. Si no se especifica el tipo de respuesta, se asigna el tipo DEFAULT_5XX .
INVALID_API_KEY	403	La respuesta de gateway para una clave de API no válida enviada para un método que requiere una clave de API. Si no se especifica el tipo de respuesta, se asigna el tipo DEFAULT_4XX .
INVALID_SIGNATURE	403	La respuesta de gateway para un error de firma de AWS no válida. Si no se especifica el tipo de respuesta, se asigna el tipo DEFAULT_4XX .
MISSING_AUTHENTICATION_TOKEN	403	La respuesta de gateway para un error de token de autenticación que falta, incluidos los casos en los que el cliente intenta invocar un método o recurso de API no admitido. Si no se especifica el tipo de respuesta, se asigna el tipo DEFAULT_4XX .

Tipo de respuesta de gateway	Código de estado predeterminado	Descripción
QUOTA_EXCEEDED	429	La respuesta de gateway para el error de cuota del plan de uso superada. Si no se especifica el tipo de respuesta, se asigna el tipo DEFAULT_4XX .
REQUEST_TOO_LARGE	413	La respuesta de gateway para el error de solicitud demasiado grande. Si no se especifica el tipo de respuesta, se asigna a: HTTP content length exceeded 10485760 bytes.
RESOURCE_NOT_FOUND	404	La respuesta de gateway cuando API Gateway no puede encontrar el recurso especificado después de que la solicitud de API supera la autenticación y autorización, excepto para la autenticación y autorización de la clave de API. Si no se especifica el tipo de respuesta, se asigna el tipo DEFAULT_4XX .
THROTTLED	429	La respuesta de gateway cuando se exceden los límites de solicitudes de nivel de plan, método, etapa o cuenta. Si no se especifica el tipo de respuesta, se asigna el tipo DEFAULT_4XX .

Tipo de respuesta de gateway	Código de estado predeterminado	Descripción
UNAUTHORIZED	401	La respuesta de gateway cuando el autorizador de Amazon Cognito o personalizado no puede autenticar al intermediario.
UNSUPPORTED_MEDIA_TYPE	415	La respuesta de gateway cuando una carga es de un tipo de medio no admitido, si se ha habilitado el comportamiento de paso a través estricto. Si no se especifica el tipo de respuesta, se asigna el tipo DEFAULT_4XX .
WAF_FILTERED	403	La respuesta de gateway cuando bloquea una solicitu AWS WAF. Si no se especifica el tipo de respuesta, se asigna el tipo DEFAULT_4XX . **Note** Las respuestas personalizadas de AWS WAF tienen prioridad sobre las respuestas de gateway personalizadas.

CORS para las API de REST en API Gateway

Intercambio de Recursos de Origen Cruzado (CORS) es una característica de seguridad del navegador que restringe las solicitudes HTTP de origen cruzado que se inician desde secuencias de comandos que se ejecutan en el navegador. Para obtener más información, consulte ¿Qué es el CORS?

Determinar si se habilita la compatibilidad con CORS

Una solicitud HTTP de origen cruzado es una que se hace para:

- Un dominio diferente (por ejemplo, de `example.com` a `amazondomains.com`)
- Un subdominio diferente (por ejemplo, de `example.com` a `petstore.example.com`)
- Un puerto diferente (por ejemplo, de `example.com` a `example.com:10777`)
- Un protocolo diferente (por ejemplo, de `https://example.com` a `http://example.com`)

Si no puede acceder a la API y recibe un mensaje de error que contiene `Cross-Origin Request Blocked`, es posible que deba habilitar CORS.

Las solicitudes HTTP de origen cruzado se pueden dividir en dos tipos: solicitudes simples y solicitudes no simples.

Habilitar CORS para una solicitud sencilla

Una solicitud HTTP es simple si se cumplen todas las condiciones siguientes:

- Se emite contra un recurso de API que permite únicamente solicitudes `GET`, `HEAD` y `POST`.
- Si se trata de una solicitud de método `POST`, debe incluir un encabezado `Origin`.
- El tipo de contenido de la carga de la solicitud es `text/plain`, `multipart/form-data` o `application/x-www-form-urlencoded`.
- La solicitud no contiene encabezados personalizados.
- Cualquier requisito adicional enumerado en la documentación de CORS de Mozilla para solicitudes simples.

Para solicitudes simples de método `POST` de origen cruzado, la respuesta del recurso debe incluir el encabezado `Access-Control-Allow-Origin: '*'` o `Access-Control-Allow-Origin:'origin'`.

Todas los demás solicitudes HTTP de origen cruzado son solicitudes no simples.

Habilitar CORS para una solicitud no sencilla

Si los recursos de su API reciben solicitudes no sencillas, deberá habilitar la compatibilidad con CORS adicional en función del tipo de integración.

Habilitar CORS para integraciones que no sean de proxy

Para estas integraciones, el protocolo CORS requiere que el navegador envíe una solicitud preliminar al servidor y espere la aprobación (o una solicitud de credenciales) desde el servidor antes de enviar la solicitud real. Debe configurar su API para enviar una respuesta adecuada a la solicitud de verificación previa.

Para crear una respuesta con comprobación previa:

1. Cree un método `OPTIONS` con una integración simulada.
2. Añada los siguientes encabezados de respuesta a la respuesta del método 200:
   - `Access-Control-Allow-Headers`
   - `Access-Control-Allow-Methods`
   - `Access-Control-Allow-Origin`
3. Configuración del comportamiento de acceso directo de integración a `NEVER`. En este caso, la solicitud del método de un tipo de contenido sin asignar se rechazará con una respuesta HTTP 415 Unsupported Media Type. Para obtener más información, consulte Comportamiento de solicitud de método para cargas útiles sin plantillas de asignación para las API de REST en API Gateway.
4. Introduzca valores para los encabezados de las respuestas. Para permitir todos los orígenes, todos los métodos y los encabezados comunes, utilice los siguientes valores de encabezado:
   - `Access-Control-Allow-Headers: 'Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token'`
   - `Access-Control-Allow-Methods: 'DELETE,GET,HEAD,OPTIONS,PUT,POST,PATCH'`
   - `Access-Control-Allow-Origin: '*'`

Tras crear la solicitud con comprobación previa, debe devolver el encabezado `Access-Control-Allow-Origin: '*'` o `Access-Control-Allow-Origin:'origin'` de todos los métodos habilitados para CORS para al menos las 200 respuestas.

Habilitar CORS para integraciones que no sean de proxy mediante Consola de administración de AWS

Puede utilizar la Consola de administración de AWS para habilitar CORS. API Gateway crea un método `OPTIONS` y agrega el encabezado `Access-Control-Allow-Origin` a las respuestas de integración de métodos existentes. Esto no siempre funciona y, a veces, es necesario modificar manualmente la respuesta de integración para que devuelva el encabezado `Access-Control-Allow-Origin` de todos los métodos compatibles con CORS de al menos las 200 respuestas.

Si tiene tipos de medios binarios configurados como `*/*` para la API, cuando API Gateway crea un método `OPTIONS`, cambie el `contentHandling` a `CONVERT_TO_TEXT`.

El siguiente comando update-integration cambia el `contentHandling` a `CONVERT_TO_TEXT` para una solicitud de integración:

```
aws apigateway update-integration \
  --rest-api-id abc123 \
  --resource-id aaa111 \
  --http-method OPTIONS \
  --patch-operations op='replace',path='/contentHandling',value='CONVERT_TO_TEXT'
```

El siguiente comando update-integration-response cambia el `contentHandling` a `CONVERT_TO_TEXT` para una respuesta de integración:

```
aws apigateway update-integration-response \
  --rest-api-id abc123 \
  --resource-id aaa111 \
  --http-method OPTIONS \
  --status-code 200 \
  --patch-operations op='replace',path='/contentHandling',value='CONVERT_TO_TEXT'
```

Habilitar la compatibilidad con CORS para integraciones de proxy

En el caso de una integración de proxy Lambda o de proxy HTTP, el backend es responsable de devolver los encabezados `Access-Control-Allow-Origin`, `Access-Control-Allow-Methods` y `Access-Control-Allow-Headers`, ya que una integración de proxy no devuelve una respuesta de integración.

Las siguientes funciones de ejemplo de Lambda devuelven los encabezados CORS necesarios:

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!')
    }
```

Temas

- Habilitar CORS en un recurso mediante la consola de API Gateway
- Habilitar CORS en un recurso mediante la característica Importar API de API Gateway
- Prueba de CORS para una API en API Gateway

Habilitar CORS en un recurso mediante la consola de API Gateway

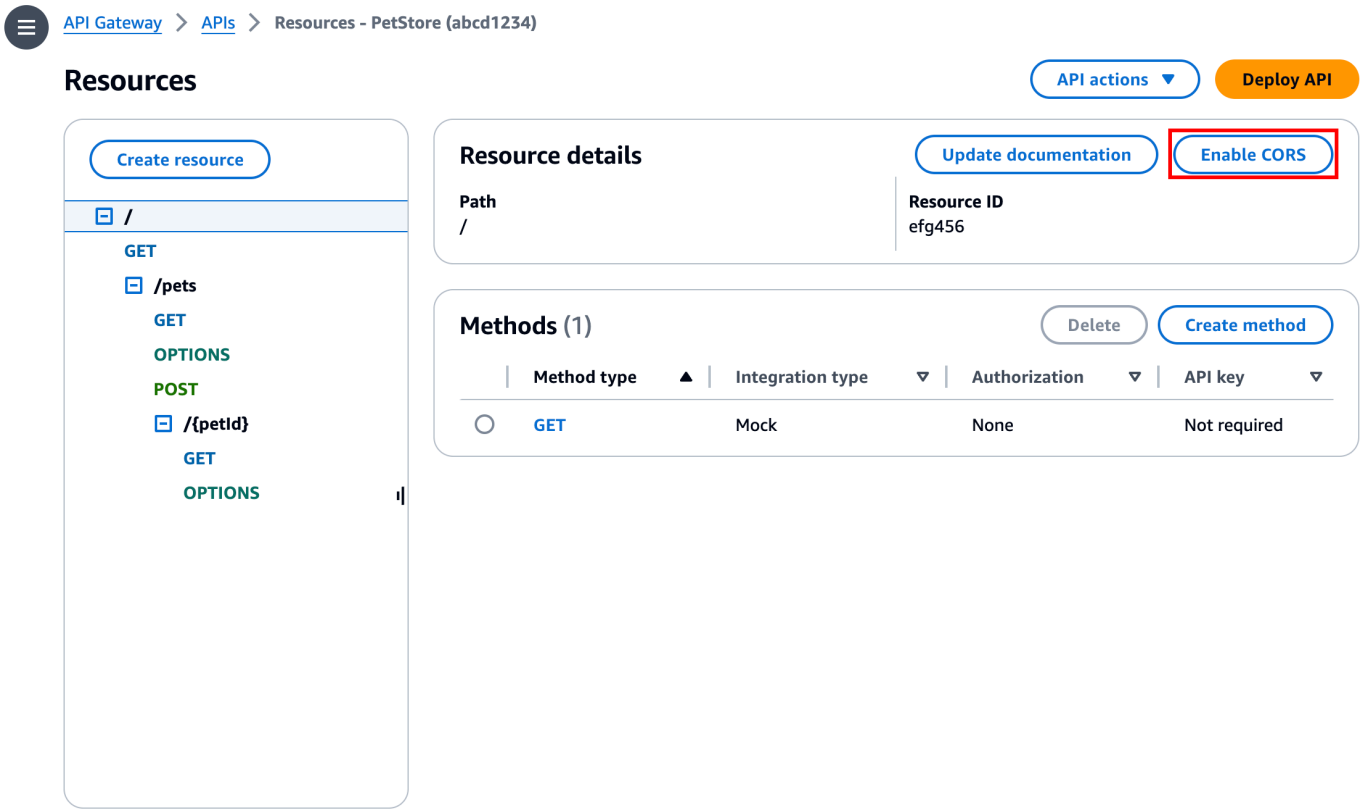
Puede utilizar la consola de API Gateway para habilitar el soporte de CORS con uno o todos los métodos en un recurso de la API REST que ha creado. Después de habilitar la compatibilidad con

COR, defina el comportamiento de acceso directo de integración como NEVER. En este caso, la solicitud del método de un tipo de contenido sin asignar se rechazará con una respuesta HTTP 415 Unsupported Media Type. Para obtener más información, consulte Comportamiento de solicitud de método para cargas útiles sin plantillas de asignación para las API de REST en API Gateway

> **Important**
>
> Los recursos pueden contener recursos secundarios. La habilitación del soporte de CORS para un recurso y sus métodos no lo habilita de forma recursiva para recursos secundarios y sus métodos.

Para habilitar soporte de CORS en un recurso de API de REST

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Elegir una API.
3. Elija un recurso en Resources (Recursos).
4. En la sección Detalles del recurso, elija Habilitar CORS.



5. En el cuadro Habilitar CORS, haga lo siguiente:

   a. (Opcional) Si creó una respuesta de puerta de enlace personalizada y desea habilitar la compatibilidad con CORS para una respuesta, seleccione una respuesta de puerta de enlace.

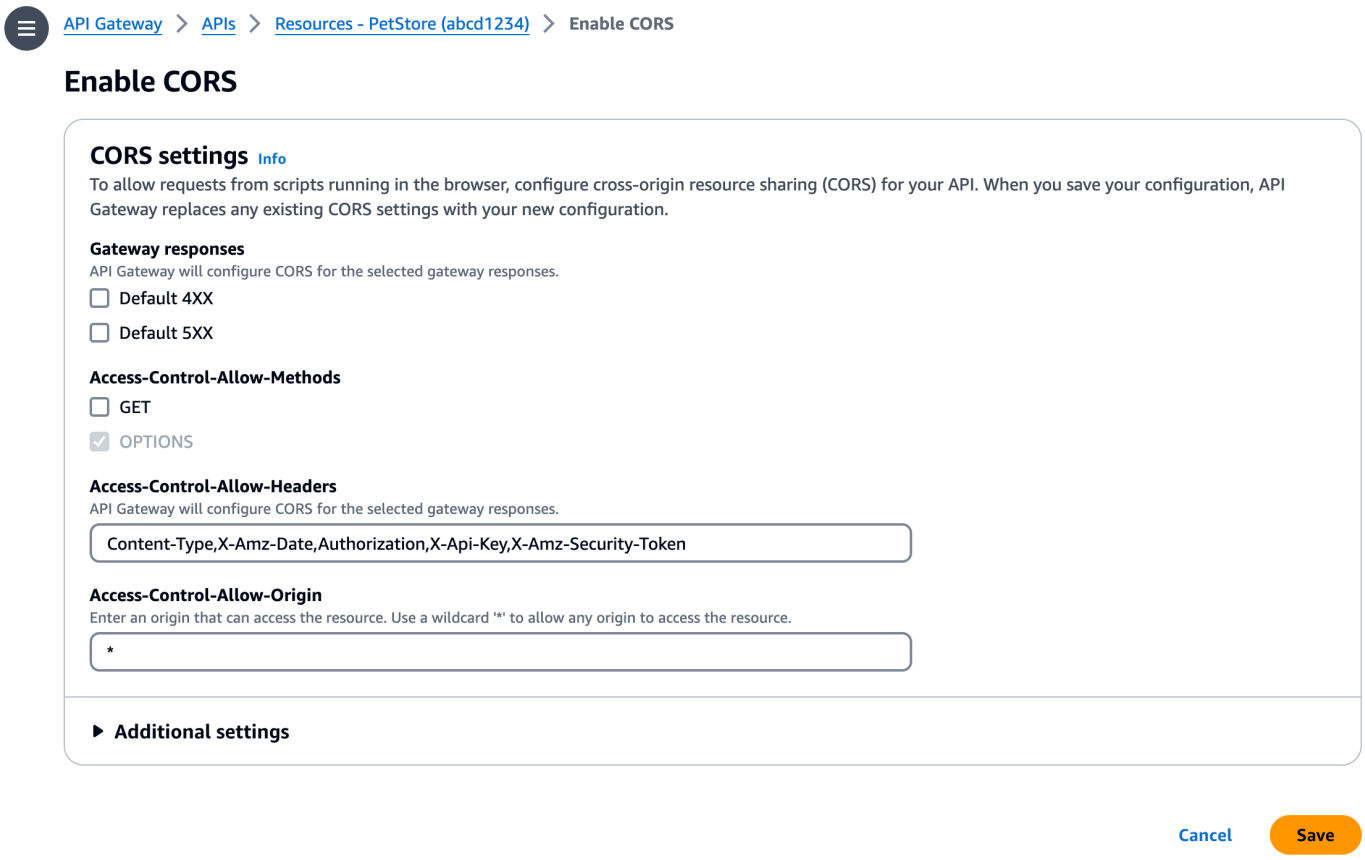
   b. Seleccione cada método para habilitar la compatibilidad con CORS. El método `OPTION` debe tener CORS habilitado.

      Si habilita la compatibilidad con CORS para un método `ANY`, se habilita CORS para todos los métodos.

   c. En el campo de entrada Access-Control-Allow-Headers, escriba una cadena estática de una lista separada por comas de encabezados que el cliente debe enviar en la solicitud real del recurso. Utilice la lista de encabezados proporcionada por la consola `'Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token'` o especifique sus propios encabezados.

   d. Utilice el valor proporcionado por la consola `'*'` como valor de encabezado Access-Control-Allow-Origin para permitir solicitudes de acceso de todos los orígenes o especifique un origen al que se permite acceder al recurso.

   e. Seleccione Save.



> **Important**
>
> Al aplicar las instrucciones anteriores al método ANY en una integración de proxy, no se establecerán los encabezados de CORS correspondientes. En su lugar, el backend debe devolver los encabezados CORS correspondientes, como por ejemplo `Access-Control-Allow-Origin`.

Una vez habilitado CORS en el método `GET`, se añade un método `OPTIONS` al recurso si no aún no se ha añadido. La respuesta `200` del método `OPTIONS` se configura automáticamente para devolver los tres encabezados `Access-Control-Allow-*` para el protocolo preliminar. Asimismo, el método (`GET`) real también está configurado de forma predeterminada para devolver el encabezado `Access-Control-Allow-Origin` en su respuesta 200. Para otros tipos de respuestas, tendrá que configurarlas manualmente para que devuelvan el encabezado `Access-Control-Allow-Origin'` con '*' un orígenes específicos, si no desea recibir el error `Cross-origin access`.

Después de habilitar la compatibilidad con CORS en su recurso, debe implementar o volver a implementar la API para que la nueva configuración surta efecto. Para obtener más información, consulte the section called “Creación de una implementación”.

> **Note**
>
> Si no puede habilitar la compatibilidad con CORS en el recurso después de seguir el procedimiento, le recomendamos que compare la configuración de CORS con el recurso `/pets` de la API de ejemplo. Para obtener información sobre cómo crear la API de ejemplo, consulte the section called “Tutorial: Crear una API de REST importando un ejemplo”.

Habilitar CORS en un recurso mediante la característica Importar API de API Gateway

Si utiliza la característica Importar API de API Gateway, puede configurar la compatibilidad de CORS con un archivo de OpenAPI. En primer lugar, debe definir un método `OPTIONS` en el recurso que devuelva los encabezados necesarios.

> **Note**
>
> Los navegadores web esperan que los encabezados Access-Control-Allow-Headers y Access-Control-Allow-Origin estén configurados en cada método de API que acepta solicitudes de CORS. Además, algunos navegadores primero realizan una solicitud HTTP a un método `OPTIONS` en el mismo recurso y, a continuación, esperan recibir los mismos encabezados.

Método **`Options`** de ejemplo

En el siguiente ejemplo, se crea un método `OPTIONS` para una integración simulada.

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"
```

Una vez que haya configurado el método OPTIONS para su recurso, puede añadir los encabezados necesarios a los otros métodos del mismo recurso necesarios para aceptar solicitudes de CORS.

1. Declare Access-Control-Allow-Origin y Headers (Encabezados) en los tipos de respuesta.

   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. En la etiqueta `x-amazon-apigateway-integration`, configure la asignación de esos encabezados a los valores estáticos:

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 : "'*'"
```

API de ejemplo

En el siguiente ejemplo, se crea una API completa con un método `OPTIONS` y un método `GET` con una integración `HTTP`.

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"
```

Prueba de CORS para una API en API Gateway

Puede probar la configuración de CORS de su API invocando su API y comprobando los encabezados de CORS en la respuesta. El siguiente comando `curl` envía una solicitud OPTIONS a una API implementada.

```
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
```

Los encabezados `Access-Control-Allow-Origin`, `Access-Control-Allow-Headers` y `Access-Control-Allow-Methods` de la respuesta muestran que la API admite CORS. Para obtener más información, consulte CORS para las API de REST en API Gateway.

Tipos de medios binarios para las API de REST en API Gateway

En API Gateway, la solicitud y respuesta de la API tienen una carga de texto o binaria. Una carga de texto es una cadena JSON codificada en `UTF-8`. Una carga binaria es cualquier otra cosa distinta de una carga de texto. La carga binaria puede ser, por ejemplo, un archivo JPEG, un archivo GZip o un archivo XML. La configuración de la API necesaria para admitir medios binarios depende de si la API utiliza integraciones de proxy o que no son de proxy.

Si utiliza una integración de proxy con transmisión de respuesta de carga útil, no necesita configurar los tipos de medios binarios. Para obtener más información, consulte the section called “Transmisión de la respuesta de integración para las integraciones de proxy”.

AWS LambdaIntegraciones proxy de

Para tratar cargas útiles binarias para integraciones proxy de AWS Lambda, debe codificar a base64 la respuesta de su función. También debe configurar binaryMediaTypes para su API. La configuración `binaryMediaTypes` de su API es una lista de tipos de contenido que su API trata como datos binarios. Los tipos de medios binarios de ejemplo incluyen `image/png` o `application/octet-stream`. Puede utilizar el carácter comodín (`*`) para cubrir varios tipos de medios.

API Gateway utiliza el primer encabezado `Accept` de los clientes para determinar si una respuesta debe devolver medios binarios. Para devolver medios binarios cuando no se puede controlar el orden de los valores de encabezado `Accept`, como las solicitudes de un navegador, establezca los tipos de medios binarios de la API en `*/*`.

Para ver el código de ejemplo, consulte the section called “Devolución de medios binarios desde una integración de proxy de Lambda en API Gateway”.

Si utiliza una integración de proxy de Lambda con transmisión de respuesta de carga útil, no necesita configurar los tipos de medios binarios. Para obtener más información, consulte the section called “Configuración de una integración de proxy de Lambda con transmisión de respuesta de carga útil”.

Integraciones que no son de proxy

Para tratar cargas binarias para integraciones que no son de proxy, agregue los tipos de medios a la lista binaryMediaTypes del recurso `RestApi`. La configuración `binaryMediaTypes` de su API es una lista de tipos de contenido que su API trata como datos binarios. Como alternativa, puede establecer las propiedades contentHandling en los recursos Integration e IntegrationResponse. El valor `contentHandling` puede ser `CONVERT_TO_BINARY`, `CONVERT_TO_TEXT` o no estar definido.

> **Note**
>
> En el caso de las integraciones de `MOCK` o privadas, no se permite establecer las propiedades `contentHandling` en la Consola de administración de AWS. Debe usar la AWS CLI, CloudFormation o un SDK para establecer las propiedades `contentHandling`.

En función del valor de `contentHandling` y de si el encabezado `Content-Type` de la respuesta o el encabezado `Accept` de la solicitud coinciden con una entrada de la lista `binaryMediaTypes`, API Gateway puede codificar los bytes binarios sin formato como una cadena codificada en base64, descodificar una cadena codificada en base64 en sus bytes sin formato o transferir el cuerpo sin realizar ninguna modificación.

Debe configurar la API como se indica a continuación para que su API de API Gateway admita cargas binarias:

- Agregue los tipos de medios binarios que desee a la lista `binaryMediaTypes` en el recurso RestApi. Si esta propiedad y la `contentHandling` propiedad no se definen, las cargas se administran como cadenas JSON codificadas en UTF-8.
- Diríjase a la propiedad `contentHandling` del recurso Integration.
  - Para que la carga de solicitud se convierta de una cadena codificada en base64 a su blob binario, establezca la propiedad en `CONVERT_TO_BINARY`.
  - Para que la carga de solicitud se convierta de un blob binario a una cadena codificada en base64, establezca la propiedad en `CONVERT_TO_TEXT`.
  - Para pasar la carga útil sin modificaciones, deje la propiedad sin definir. Para pasar una carga binaria sin modificaciones, también debe asegurarse de que `Content-Type` coincide con una de las entradas de `binaryMediaTypes` y que los comportamientos del acceso directo están habilitados para la API.
- Establezca la propiedad `contentHandling` del recurso IntegrationResponse. La propiedad `contentHandling`, el encabezado `Accept` en las solicitudes de cliente y los `binaryMediaTypes` de su API combinados determinan cómo API Gateway trata las conversiones de los tipos de contenido. Para obtener información, consulte the section called “Conversiones de tipo de contenido en API Gateway”.

> **Important**
>
> Cuando una solicitud contiene varios tipos de medios en su encabezado `Accept`, API Gateway solo respeta el primer tipo de medio `Accept`. Si no puede controlar el orden de los tipos de medios `Accept` y el tipo de medio del contenido binario no sea el primero de la lista, agregue el primer tipo de medio `Accept` en la lista `binaryMediaTypes` de la API. API Gateway gestiona todos los tipos de contenido de esta lista como binarios.
> Por ejemplo, para enviar un archivo JPEG utilizando un elemento `<img>` en un navegador, el navegador puede enviar `Accept:image/webp,image/*,*/*;q=0.8` en una solicitud. Al añadir `image/webp` a la lista `binaryMediaTypes`, el punto de conexión recibe el archivo JPEG como binario.

Para obtener información detallada sobre cómo API Gateway trata el texto y las cargas binarias, consulte Conversiones de tipo de contenido en API Gateway.

Conversiones de tipo de contenido en API Gateway

La combinación de los `binaryMediaTypes` de su API, los encabezados de las solicitudes de cliente y la propiedad `contentHandling` de integración determinan cómo API Gateway codifica las cargas útiles.

La siguiente tabla muestra cómo API Gateway convierte la carga útil de solicitud para configuraciones específicas del encabezado `Content-Type` de una solicitud, la lista `binaryMediaTypes` de un recurso RestApi y el valor de la propiedad `contentHandling` del recurso Integración.

Carga de solicitud de método	Encabezado **`Content-Type`** de solicitud	**`binaryMediaTypes`**	**`contentHandling`**	Carga de solicitud de integración
Datos de texto	Cualquier tipo de datos	Sin definir	Sin definir	Cadena codificada en UTF8
Datos de texto	Cualquier tipo de datos	Sin definir	`CONVERT_TO_BINARY`	Blob binario descodificado en Base64

Carga de solicitud de método	Encabezado **`Content-Type`** de solicitud	**`binaryMediaTypes`**	**`contentHandling`**	Carga de solicitud de integración
Datos de texto	Cualquier tipo de datos	Sin definir	`CONVERT_TO_TEXT`	Cadena codificada en UTF8
Datos de texto	Un tipo de datos de texto	Establecida con tipos de medios coincidentes	Sin definir	Datos de texto
Datos de texto	Un tipo de datos de texto	Establecida con tipos de medios coincidentes	`CONVERT_TO_BINARY`	Blob binario descodificado en Base64
Datos de texto	Un tipo de datos de texto	Establecida con tipos de medios coincidentes	`CONVERT_TO_TEXT`	Datos de texto
Datos binarios	Un tipo de datos binarios	Establecida con tipos de medios coincidentes	Sin definir	Datos binarios
Datos binarios	Un tipo de datos binarios	Establecida con tipos de medios coincidentes	`CONVERT_TO_BINARY`	Datos binarios
Datos binarios	Un tipo de datos binarios	Establecida con tipos de medios coincidentes	`CONVERT_TO_TEXT`	Cadena codificada en Base64

La siguiente tabla muestra cómo API Gateway convierte la carga útil de respuesta para configuraciones específicas del encabezado `Accept` de una solicitud, la lista `binaryMediaTypes` de un recurso RestApi y el valor de la propiedad `contentHandling` del recurso Respuesta de integración.

> **Important**
>
> Cuando una solicitud contiene varios tipos de medios en su encabezado `Accept`, API Gateway solo respeta el primer tipo de medio `Accept`. Si no puede controlar el orden de los tipos de medios `Accept` y el tipo de medio del contenido binario no sea el primero de la lista, agregue el primer tipo de medio `Accept` en la lista `binaryMediaTypes` de la API. API Gateway gestiona todos los tipos de contenido de esta lista como binarios.
> Por ejemplo, para enviar un archivo JPEG utilizando un elemento `<img>` en un navegador, el navegador puede enviar `Accept:image/webp,image/*,*/*;q=0.8` en una solicitud. Al añadir `image/webp` a la lista `binaryMediaTypes`, el punto de conexión recibe el archivo JPEG como binario.

Carga de respuesta de integración	Encabezado **Accept** de solicitud	**binaryMediaTypes**	**contentHandling**	Carga de respuesta de método
Datos de texto o binarios	Un tipo de texto	Sin definir	Sin definir	Cadena codificada en UTF8
Datos de texto o binarios	Un tipo de texto	Sin definir	`CONVERT_TO_BINARY`	Blob descodificado en Base64
Datos de texto o binarios	Un tipo de texto	Sin definir	`CONVERT_TO_TEXT`	Cadena codificada en UTF8
Datos de texto	Un tipo de texto	Establecida con tipos de medios coincidentes	Sin definir	Datos de texto
Datos de texto	Un tipo de texto	Establecida con tipos de medios coincidentes	`CONVERT_TO_BINARY`	Blob descodificado en Base64

Carga de respuesta de integración	Encabezado **Accept** de solicitud	**binaryMediaTypes**	**contentHandling**	Carga de respuesta de método
Datos de texto	Un tipo de texto	Establecida con tipos de medios coincidentes	`CONVERT_TO_TEXT`	Cadena codificada en UTF8
Datos de texto	Un tipo binario	Establecida con tipos de medios coincidentes	Sin definir	Blob descodificado en Base64
Datos de texto	Un tipo binario	Establecida con tipos de medios coincidentes	`CONVERT_TO_BINARY`	Blob descodificado en Base64
Datos de texto	Un tipo binario	Establecida con tipos de medios coincidentes	`CONVERT_TO_TEXT`	Cadena codificada en UTF8
Datos binarios	Un tipo de texto	Establecida con tipos de medios coincidentes	Sin definir	Cadena codificada en Base64
Datos binarios	Un tipo de texto	Establecida con tipos de medios coincidentes	`CONVERT_TO_BINARY`	Datos binarios
Datos binarios	Un tipo de texto	Establecida con tipos de medios coincidentes	`CONVERT_TO_TEXT`	Cadena codificada en Base64
Datos binarios	Un tipo binario	Establecida con tipos de medios coincidentes	Sin definir	Datos binarios
Datos binarios	Un tipo binario	Establecida con tipos de medios coincidentes	`CONVERT_TO_BINARY`	Datos binarios

Carga de respuesta de integración	Encabezado o **`Accept`** de solicitud	**`binaryMediaTypes`**	**`contentHandling`**	Carga de respuesta de método
Datos binarios	Un tipo binario	Establecida con tipos de medios coincidentes	`CONVERT_TO_TEXT`	Cadena codificada en Base64

Cuando se convierte una carga de texto en un blob binario, API Gateway asume que los datos de texto son una cadena codificada en base64 y envía los datos binarios como un blob descodificado en base64. Si la conversión produce un error, devuelve una respuesta `500`, que indica un error de configuración de la API. No tiene que proporcionar una plantilla de asignación para este tipo de conversión, pero sí debe habilitar los comportamientos de paso a través en la API.

Al convertir una carga binaria en una cadena de texto, API Gateway siempre aplica una codificación en base64 a los datos binarios. Puede definir una plantilla de asignación para este tipo de carga, pero solo puede tener acceso a la cadena codificada en base64 en la plantilla de asignación a través de `$input.body`, tal y como se muestra en el fragmento siguiente de una plantilla de asignación de ejemplo.

```
{
    "data": "$input.body"
}
```

Para que se transfiera la carga binaria sin modificaciones, debe habilitar los comportamientos de paso a través en la API.

Habilitar la compatibilidad con datos binarios mediante la consola de API Gateway

En esta sección se explica cómo habilitar la compatibilidad con datos binarios en la consola de API Gateway. A modo de ejemplo, usaremos una API que está integrada con Amazon S3. Nos centraremos en las tareas para establecer tipos de medios compatibles y para especificar cómo debe controlarse la carga. Para obtener información detallada sobre cómo crear una API integrada con Amazon S3, consulte Tutorial: Creación de una API de REST como proxy de Amazon S3.

Para habilitar la compatibilidad con datos binarios utilizando la consola de API Gateway

1. Establezca tipos de medios binarios para la API:

a. Cree una nueva API o elija una API existente. En este ejemplo, asignaremos a la API el nombre de `FileMan`.

b. Bajo la API seleccionada, en el panel de navegación principal, seleccione Configuración de API.

c. En el panel Configuración de API, elija Administrar tipos de medios en la sección Tipos de medios binarios.

d. Seleccione Añadir tipo de medio binario.

e. Escriba un tipo de medio necesario (por ejemplo, **`image/png`**) en el campo de texto. Si es necesario, repita este paso para añadir más tipos de medios. Para admitir todos los tipos de medios binarios, especifique `*/*`.

f. Seleccione Save changes (Guardar cambios).

2. Establezca cómo se administran las cargas de mensajes para el método de API:

a. Cree un nuevo recurso o elija uno existente en la API. Para este ejemplo, usaremos el recurso `/{folder}/{item}`.

b. Cree un nuevo método o elija uno existente en el recurso. Como ejemplo, usaremos el método `GET /{folder}/{item}` integrado con la acción `Object GET` en Amazon S3.

c. En Tratamiento de contenido, elija una opción.



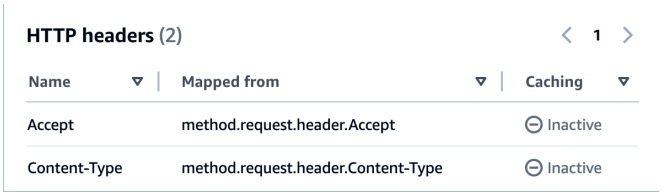
Elija Passthrough (Paso a través) si no desea convertir el cuerpo cuando el cliente y el backend acepten el mismo formato binario. Elija Convertir en texto para convertir el cuerpo

binario en una cadena codificada en base64 cuando, por ejemplo, el backend requiera que una carga de solicitud binaria se pase como una propiedad JSON. Y elija Convertir en binario cuando el cliente envíe una cadena codificada en base64 y el backend requiera el formato binario original, o cuando el punto de conexión devuelva una cadena codificada en base64 y el cliente acepte únicamente la salida binaria.

d. En Acceso directo de cuerpo de la solicitud, elija Cuando no hay plantillas definidas (recomendado) para activar el comportamiento de acceso directo en el cuerpo de la solicitud.

   También puede elegir Nunca. Esto significa que la API rechazará los datos con tipos de contenido que no tengan una plantilla de mapeo.

e. Conserve el encabezado `Accept` de la solicitud entrante en la solicitud de integración. Debe hacerlo si ha establecido `contentHandling` en `passthrough` y desea invalidar esa configuración en tiempo de ejecución.



f. Para la conversión en texto, defina una plantilla de asignación para aplicar a los datos binarios codificados en base64 el formato requerido.

   Este es un ejemplo de plantilla de mapeo para convertirla en texto:

```
{
  "operation": "thumbnail",
  "base64Image": "$input.body"
}
```

   El formato de esta plantilla de asignación depende de los requisitos de punto de conexión de la entrada.

g. Seleccione Save.

Habilitar la compatibilidad con datos binarios mediante la API Gateway REST API

Las siguientes tareas muestran cómo habilitar la compatibilidad con datos binarios mediante llamadas a la API REST de API Gateway.

Temas

- Añadir y actualizar tipos de medios binarios compatibles en una API
- Configurar las conversiones de carga de solicitud
- Configurar conversiones de carga de respuesta
- Convertir datos binarios en datos de texto
- Convertir datos de texto en una carga binaria
- Transferir una carga binaria

Añadir y actualizar tipos de medios binarios compatibles en una API

Para habilitar API Gateway para que admita un nuevo tipo de medios binarios, debe agregar el tipo de medios binarios a la lista `binaryMediaTypes` del recurso `RestApi`. Por ejemplo, para que API Gateway admita imágenes JPEG, envíe una solicitud `PATCH` al recurso `RestApi`:

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/binaryMediaTypes/image~1jpeg"
  }
 ]
}
```

La especificación del tipo MIME de `image/jpeg` que forma parte del valor de la propiedad `path` se ha incluido en una secuencia de escape: `image~1jpeg`.

Para actualizar los tipos de medios binarios compatibles, sustituya o elimine el tipo de medios en la lista `binaryMediaTypes` del recurso `RestApi`. Por ejemplo, para cambiar la compatibilidad binaria de archivos JPEG a bytes sin formato, envíe una solicitud `PATCH` al recurso `RestApi`, tal y como se indica a continuación:

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [{
    "op" : "replace",
```

```
    "path" : "/binaryMediaTypes/image~1jpeg",
    "value" : "application/octet-stream"
  },
  {
    "op" : "remove",
    "path" : "/binaryMediaTypes/image~1jpeg"
  }]
}
```

Configurar las conversiones de carga de solicitud

Si el punto de conexión requiere una entrada binaria, establezca la propiedad `contentHandling` del recurso `Integration` en `CONVERT_TO_BINARY`. Para ello, envíe una solicitud `PATCH` de la siguiente manera:

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/integration

{
  "patchOperations" : [ {
    "op" : "replace",
    "path" : "/contentHandling",
    "value" : "CONVERT_TO_BINARY"
  }]
}
```

Configurar conversiones de carga de respuesta

Si el cliente acepta el resultado como un blob binario en lugar de una carga codificada en base64 devuelta por el punto de conexión, establezca la propiedad `contentHandling` del recurso `IntegrationResponse` en `CONVERT_TO_BINARY`. Para ello, envíe una solicitud `PATCH`, de la siguiente manera:

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/integration/
responses/<status_code>

{
  "patchOperations" : [ {
    "op" : "replace",
    "path" : "/contentHandling",
    "value" : "CONVERT_TO_BINARY"
  }]
```

```
}
```

Convertir datos binarios en datos de texto

Para enviar datos binarios como una propiedad JSON de la entrada a AWS Lambda o Kinesis a través de API Gateway, haga lo siguiente:

1. Habilite la compatibilidad de carga binaria de la API añadiendo el nuevo tipo de medios binarios de `application/octet-stream` a la lista `binaryMediaTypes` de la API.

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/binaryMediaTypes/application~1octet-stream"
  }
 ]
}
```

2. Establezca `CONVERT_TO_TEXT` en la propiedad `contentHandling` del recurso `Integration` y proporcione una plantilla de asignación para asignar la cadena codificada en base64 de los datos binarios a una propiedad JSON. En el siguiente ejemplo, la propiedad JSON es `body` y `$input.body` contiene la cadena codificada en base64.

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/
integration

{
  "patchOperations" : [
    {
      "op" : "replace",
      "path" : "/contentHandling",
      "value" : "CONVERT_TO_TEXT"
    },
    {
      "op" : "add",
      "path" : "/requestTemplates/application~1octet-stream",
      "value" : "{\"body\": \"$input.body\"}"
    }
  ]
}
```

Convertir datos de texto en una carga binaria

Supongamos que tiene una función de Lambda que devuelve un archivo de imagen como una cadena codificada en base64. Para pasar esta salida binaria al cliente a través de API Gateway, haga lo siguiente:

1. Actualice la lista `binaryMediaTypes` de la API añadiendo el tipo de medios binarios de `application/octet-stream`, si aún no está en la lista.

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/binaryMediaTypes/application~1octet-stream",
  }]
}
```

2. Establezca la propiedad `contentHandling` del recurso `Integration` en `CONVERT_TO_BINARY`. No defina una plantilla de asignación. Si no define una plantilla de mapeo, API Gateway llama a la plantilla de paso a través para devolver el blob binario descodificado en base64 como el archivo de imagen al cliente.

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/
integration/responses/<status_code>

{
  "patchOperations" : [
    {
      "op" : "replace",
      "path" : "/contentHandling",
      "value" : "CONVERT_TO_BINARY"
    }
  ]
}
```

Transferir una carga binaria

Para almacenar una imagen en un bucket de Amazon S3 utilizando API Gateway, haga lo siguiente:

1. Actualice la lista `binaryMediaTypes` de la API añadiendo el tipo de medios binarios de `application/octet-stream`, si aún no está en la lista.

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/binaryMediaTypes/application~1octet-stream"
  }
 ]
}
```

2. En la propiedad `contentHandling` del recurso `Integration`, establezca `CONVERT_TO_BINARY`. Establezca `WHEN_NO_MATCH` como el valor de la propiedad `passthroughBehavior` sin definir una plantilla de asignación. Esto permite a API Gateway invocar la plantilla de acceso directo.

```
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"
    }
  ]
}
```

Importación y exportación de codificaciones de contenido para API Gateway

Para importar la lista `binaryMediaTypes` de un recurso RestApi, use la siguiente extensión de API Gateway al archivo de definición de OpenAPI de la API. La extensión también se utiliza para exportar la configuración de la API.

- Propiedad x-amazon-apigateway-binary-media-types

Para importar y exportar los valores de la propiedad `contentHandling` en un recurso `Integration` o `IntegrationResponse`, utilice las siguientes extensiones de API Gateway a las definiciones de OpenAPI:

- Objeto x-amazon-apigateway-integration
- Objeto x-amazon-apigateway-integration.response

Devolución de medios binarios desde una integración de proxy de Lambda en API Gateway

Para devolver medios binarios desde una integración de proxy de AWS Lambda, base64 codifica la respuesta de su función de Lambda. También debe configurar los tipos de medios binarios de su API. Cuando configura los tipos de medios binarios de la API, esta trata ese tipo de contenido como datos binarios. El límite de tamaño de carga es 10 MB.

> **Note**
>
> Para utilizar un navegador web para invocar una API con esta integración de ejemplo, establezca los tipos de medios binarios de la API en `*/*`. API Gateway utiliza el primer encabezado `Accept` de los clientes para determinar si una respuesta debe devolver medios binarios. Para devolver medios binarios cuando no puede controlar el orden de los valores de encabezado `Accept`, como las solicitudes de un navegador, establezca los tipos de medios binarios de la API en `*/*` (para todos los tipos de contenido).

En el siguiente ejemplo, la función de Lambda puede devolver una imagen binaria a partir de Amazon S3 o texto a los clientes. La respuesta de la función incluye un encabezado `Content-Type` para indicar al cliente el tipo de datos que devuelve. La función establece condicionalmente la propiedad `isBase64Encoded` en su respuesta, dependiendo del tipo de datos que devuelve.

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>",
        }
```

Para obtener más información sobre los tipos de medios binarios, consulte Tipos de medios binarios para las API de REST en API Gateway.

Acceder a archivos binarios en Amazon S3 a través de una API de API Gateway

Los siguientes ejemplos muestran el archivo de OpenAPI utilizado para obtener acceso a las imágenes en Amazon S3, cómo descargar una imagen de Amazon S3 y cómo cargar una imagen en Amazon S3.

Temas

- Archivo de OpenAPI de una API de ejemplo para obtener acceso a imágenes en Amazon S3
- Descargar una imagen de Amazon S3
- Cargar una imagen en Amazon S3

Archivo de OpenAPI de una API de ejemplo para obtener acceso a imágenes en Amazon S3

El siguiente archivo de OpenAPI muestra una API de ejemplo que ilustra cómo descargar un archivo de imagen de Amazon S3 y cargarlo en Amazon S3. Esta API expone los métodos `GET /s3?`

`key={file-name}` y `PUT /s3?key={file-name}` para descargar y cargar un archivo de imagen especificado. El método `GET` devuelve el archivo de imagen como una cadena codificada en base64 como parte de una salida JSON, siguiendo la plantilla de asignación suministrada, en una respuesta 200 OK. El método `PUT` toma un blob binario sin formato como entrada y devuelve una respuesta 200 OK con una carga vacía.

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"
    }
  }
}
```

Descargar una imagen de Amazon S3

Para descargar un archivo de imagen (`image.jpg`) como un blob binario desde Amazon S3:

```
GET /v1/s3?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
Content-Type: application/json
Accept: application/octet-stream
```

Una respuesta correcta tiene un aspecto similar al siguiente:

```
200 OK HTTP/1.1

[raw bytes]
```

Los bytes sin procesar se devuelven, ya que el encabezado `Accept` está establecido en un tipo de medio binario de `application/octet-stream` y la compatibilidad con los datos binarios está habilitada para la API.

Para descargar un archivo de imagen (`image.jpg`) como una cadena codificada en base64 con formato de propiedad JSON, también puede, en Amazon S3, agregar una plantilla de respuesta a la respuesta de integración 200, como se muestra en el siguiente bloque de definición de OpenAPI que aparece en negrita:

```
        "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}"
              }
            }
          },
```

La solicitud para descargar el archivo de imagen tiene un aspecto similar al siguiente:

```
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
```

Una respuesta correcta tiene un aspecto similar al siguiente:

```
200 OK HTTP/1.1

{
  "image": "W3JhdyBieXRlc10="
```

```
}
```

Cargar una imagen en Amazon S3

Para cargar un archivo de imagen (`image.jpg`) como un blob binario en Amazon S3:

```
PUT /v1/s3?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
Content-Type: application/octet-stream
Accept: application/json

[raw bytes]
```

Una respuesta correcta tiene un aspecto similar al siguiente:

```
200 OK HTTP/1.1
```

Para cargar un archivo de imagen (`image.jpg`) como una cadena codificada en base64 en Amazon S3:

```
PUT /v1/s3?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
Content-Type: application/json
Accept: application/json

W3JhdyBieXRlc10=
```

La carga de entrada debe ser una cadena codificada en base64, ya que el valor del encabezado `Content-Type` está establecido en `application/json`. Una respuesta correcta tiene un aspecto similar al siguiente:

```
200 OK HTTP/1.1
```

Acceder a archivos binarios en Lambda a través de una API de API Gateway

El siguiente ejemplo de OpenAPI ilustra cómo acceder a un archivo binario en AWS Lambda a través de una API de API Gateway. Esta API expone los métodos `GET /lambda?key={file-name}` y `PUT /lambda?key={file-name}` para descargar y cargar un archivo de imagen especificado. El método `GET` devuelve el archivo de imagen como una cadena codificada en base64 como parte de una salida JSON, siguiendo la plantilla de asignación suministrada, en una respuesta 200 OK. El

método PUT toma un blob binario sin formato como entrada y devuelve una respuesta 200 OK con una carga vacía.

Se crea la función de Lambda a la que llama la API y esta debe devolver una cadena codificada en base64 con el encabezado `Content-Type` de `application/json`.

Temas

- Archivo de OpenAPI de una API de ejemplo para obtener acceso a imágenes en Lambda
- Descargar una imagen de Lambda
- Cargar una imagen en Lambda

Archivo de OpenAPI de una API de ejemplo para obtener acceso a imágenes en Lambda

El siguiente archivo de OpenAPI muestra una API de ejemplo que ilustra cómo descargar un archivo de imagen de Lambda y cargarlo en Lambda.

OpenAPI 3.0

```
{
   "openapi": "3.0.0",
   "info": {
      "version": "2016-10-21T17:26:28Z",
      "title": "ApiName"
   },
   "paths": {
      "/lambda": {
         "get": {
            "parameters": [
               {
                  "name": "key",
                  "in": "query",
                  "required": false,
                  "schema": {
                     "type": "string"
                  }
               }
            ],
            "responses": {
               "200": {
                  "description": "200 response",
                  "content": {
```

```
                    "application/json": {
                       "schema": {
                          "$ref": "#/components/schemas/Empty"
                       }
                    }
                 }
              },
              "500": {
                 "description": "500 response"
              }
           },
           "x-amazon-apigateway-integration": {
              "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-east-1:123456789012:function:image/invocations",
              "type": "AWS",
              "credentials": "arn:aws:iam::123456789012:role/Lambda",
              "httpMethod": "POST",
              "requestTemplates": {
                 "application/json": "{\n   \"imageKey\":
 \"$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"
    }
  }
}
```

Descargar una imagen de Lambda

Para descargar un archivo de imagen (`image.jpg`) como un blob binario desde Lambda:

```
GET /v1/lambda?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
Content-Type: application/json
Accept: application/octet-stream
```

Una respuesta correcta tiene un aspecto similar al siguiente:

```
200 OK HTTP/1.1

[raw bytes]
```

Para descargar un archivo de imagen (`image.jpg`) como una cadena codificada en base64, formateada como una propiedad JSON, desde Lambda:

```
GET /v1/lambda?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
Content-Type: application/json
Accept: application/json
```

Una respuesta correcta tiene un aspecto similar al siguiente:

```
200 OK HTTP/1.1

{
  "image": "W3JhdyBieXRlc10="
}
```

Cargar una imagen en Lambda

Para cargar un archivo de imagen (`image.jpg`) como un blob binario en Lambda:

```
PUT /v1/lambda?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
Content-Type: application/octet-stream
Accept: application/json

[raw bytes]
```

Una respuesta correcta tiene un aspecto similar al siguiente:

```
200 OK
```

Para cargar un archivo de imagen (`image.jpg`) como una cadena codificada en base64 en Lambda:

```
PUT /v1/lambda?key=image.jpg HTTP/1.1
Host: abcdefghi.execute-api.us-east-1.amazonaws.com
Content-Type: application/json
Accept: application/json

W3JhdyBieXRlc10=
```

Una respuesta correcta tiene un aspecto similar al siguiente:

```
200 OK
```

Invocación de las API de REST en API Gateway

Para llamar a una API implementada, los clientes envían solicitudes a la URL del servicio del componente de API Gateway para la ejecución de API, denominado `execute-api`.

La URL base de las API de REST tiene el siguiente formato:

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

donde *`api-id`* es el identificador de la API, *`region`* es la región de AWS y *`stage`* es el nombre de la etapa de la implementación de la API.

> **Important**
>
> Antes de poder invocar una API, debe implementarla en API Gateway. Para obtener instrucciones sobre cómo implementar una API, consulte Implementación de las API de REST en API Gateway.

Temas

- Obtención de la URL de invocación de una API
- Invocación de una API
- Uso de la consola de API Gateway para probar un método de la API REST
- Uso de un SDK de Java generado por API Gateway para una API REST
- Uso de un SDK de Android generado por API Gateway para una API REST
- Uso de un SDK de JavaScript generado por API Gateway para una API REST
- Uso de un SDK de Ruby generado por API Gateway para una API REST
- Uso de un SDK de iOS generado por API Gateway para una API REST en Objective-C o Swift

Obtención de la URL de invocación de una API

Puede usar la consola, la AWS CLI o una definición de OpenAPI exportada para obtener la URL de invocación de una API.

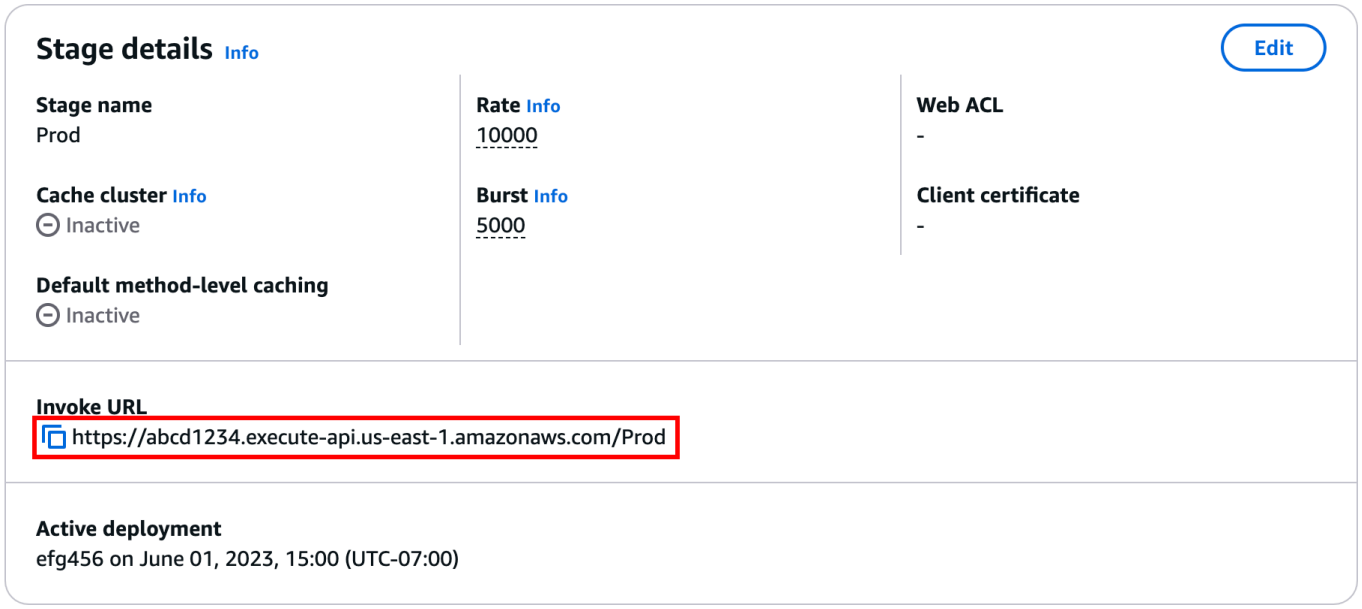
Obtención de la URL de invocación de una API con la consola

En el siguiente procedimiento se muestra cómo obtener la URL de invocación de una API en la consola de la API de REST.

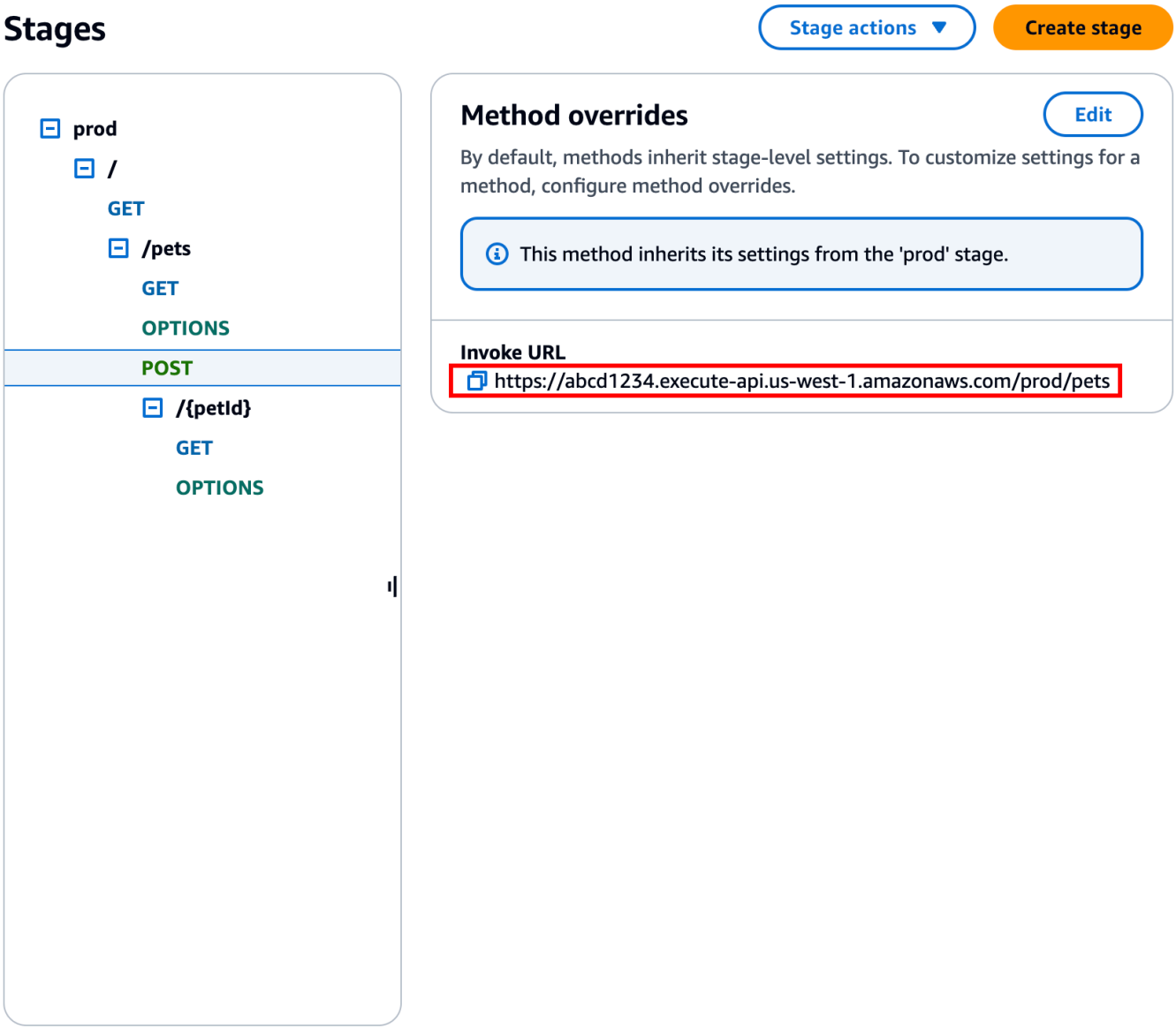
Obtención de la URL de invocación de una API con la consola de API de REST

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API implementada.
3. En el panel de navegación principal, elija Etapa.
4. En Detalles de la etapa, elija el icono de copia para copiar la URL de invocación de la API.

   Esta URL es para el recurso raíz de la API.



5. Para obtener la URL de invocación de una API para otro recurso de la API, expanda la etapa bajo el panel de navegación secundario y, a continuación, elija un método.
6. Elija el icono de copiar para copiar la URL de invocación en el nivel de recursos de la API.



Obtención de la URL de invocación de una API con la AWS CLI

En el siguiente procedimiento se muestra cómo obtener la URL de invocación de una API con la AWS CLI.

Obtención de la URL de invocación de una API con la AWS CLI

1. Utilice el siguiente comando para obtener `rest-api-id`. Este comando devuelve todos los valores `rest-api-id` de la región. Para obtener más información, consulte get-rest-apis.

```
aws apigateway get-rest-apis
```

2. Sustituya `rest-api-id` del ejemplo por su `rest-api-id`, sustituya *`{stage-name}`* del ejemplo por su *`{stage-name}`* y sustituya *`{region}`* por su región.

```
https://{restapi_id}.execute-api.{region}.amazonaws.com/{stage_name}/
```

Obtención de la URL de invocación de una API mediante el archivo de definición de OpenAPI exportado de la API

También puede crear la URL raíz combinando los campos `host` y `basePath` de un archivo de definición de OpenAPI exportado de la API. Para obtener instrucciones acerca de cómo exportar la API, consulte the section called “Exportación de una API de REST”.

Invocación de una API

Puede llamar a la API implementada mediante el navegador, curl u otras aplicaciones, como Postman.

Además, puede utilizar la consola de API Gateway para probar una llamada a la API. La prueba utiliza la característica `TestInvoke` de API Gateway, que permite probar la API antes de que se implemente la API. Para obtener más información, consulte the section called “Uso de la consola para probar un método de la API de REST”.

> **Note**
>
> Los valores de los parámetros de cadenas de consulta en una URL de invocación no pueden contener `%%`.

Invocación de una API mediante un navegador web

Si la API permite el acceso anónimo, puede utilizar cualquier navegador web para invocar cualquier método `GET`. Ingrese la URL de invocación completa en la barra de dirección del navegador.

Para otros métodos o para todas las llamadas que requieran autenticación, debe especificar una carga o firmar las solicitudes. Puede realizar esto en un script detrás de una página HTML o en una aplicación cliente mediante uno de los SDK de AWS.

Invocación de una API mediante curl

Puede usar una herramienta como curl en el terminal para llamar a la API. El siguiente comando curl de ejemplo invoca el método GET en el recurso `getUsers` de la fase `prod` de una API.

Linux or Macintosh

```
curl -X GET 'https://b123abcde4.execute-api.us-west-2.amazonaws.com/prod/getUsers'
```

Windows

```
curl -X GET "https://b123abcde4.execute-api.us-west-2.amazonaws.com/prod/getUsers"
```

Uso de la consola de API Gateway para probar un método de la API REST

Uso de la consola de API Gateway para probar un método de la API REST.

Temas

- Requisitos previos
- Probar un modelo con la consola de API Gateway

Requisitos previos

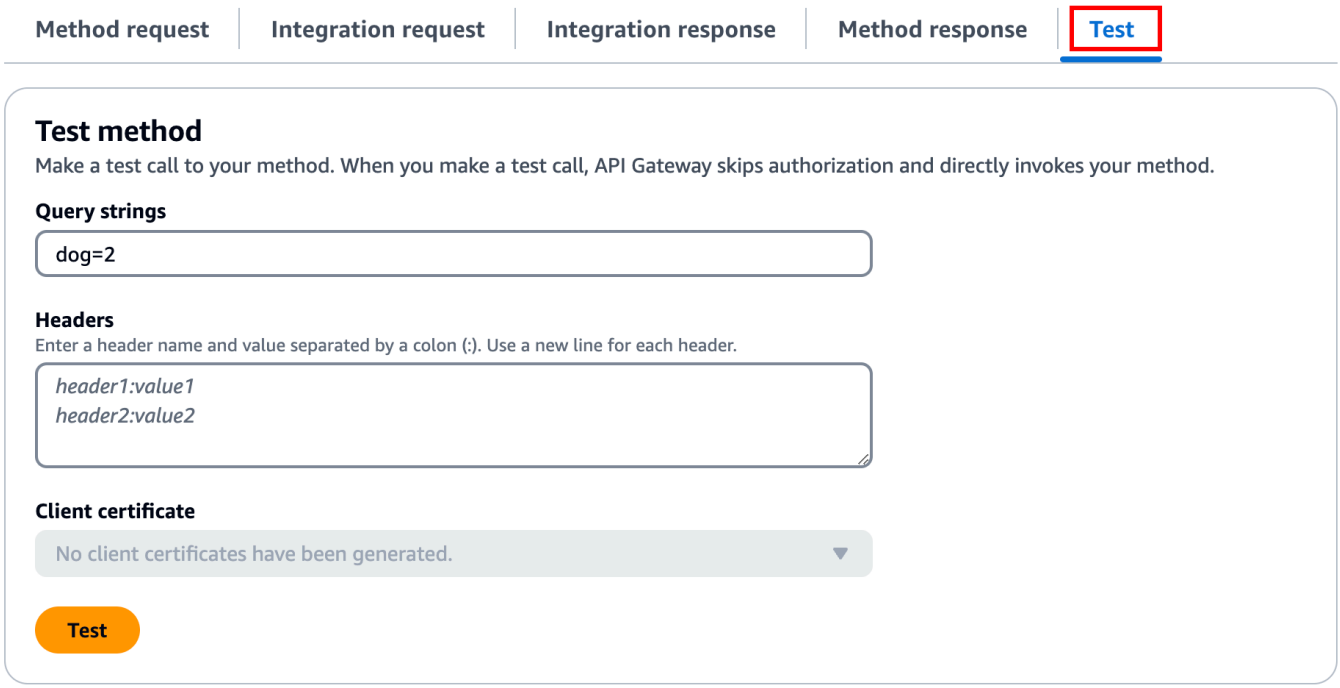
- Debe especificar la configuración de los métodos que desea probar. Siga las instrucciones en Métodos de API de REST en API Gateway.

Probar un modelo con la consola de API Gateway

> **Important**
>
> La prueba de métodos con la consola de API Gateway puede provocar cambios en los recursos que no se pueden deshacer. Probar un método con la consola API Gateway es lo mismo que llamar al método desde fuera de la consola de API Gateway. Por ejemplo, si utiliza la consola de API Gateway para llamar a un método que elimina recursos de una API, si la llamada al método se realiza correctamente, se eliminarán los recursos de la API.

Para probar un método

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST.
3. En el panel Resources (Recursos), elija el método que desea probar.
4. Elija la pestaña Prueba. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña.



   Escriba valores en cualquiera de los cuadros mostrados (como Cadenas de consulta), Encabezados y Cuerpo de solicitud). La consola incluye estos valores en la solicitud de método con el formato application/json predeterminado.

   Si necesita especificar opciones adicionales, póngase en contacto con el propietario de la API.
5. Seleccione Test (Probar). Se mostrará la siguiente información:

   - Request (Solicitud) es la ruta del recurso llamada para el método.
   - Status (Estado) es el código de estado HTTP de la respuesta.
   - Latencia (ms) es el tiempo entre la recepción de la solicitud del intermediario y la respuesta devuelta.
   - Cuerpo de respuesta es el cuerpo de la respuesta HTTP.
   - Encabezados de respuesta son los encabezados de respuesta HTTP.

> **Tip**
>
> En función del mapeo, el código de estado HTTP, el cuerpo de la respuesta y los encabezados de respuesta podrían ser diferentes de los enviados desde la función de Lambda, proxy HTTP o proxy de servicio de AWS.

- Losregistros son las entradas simuladas de Amazon CloudWatch Logs que se habrían escrito si se hubiera llamado a este método fuera de la consola de API Gateway.

> **Note**
>
> Aunque las entradas de CloudWatch Logs son simuladas, los resultados de la llamada al método son reales.

Además de utilizar la consola de API Gateway, puede utilizar la AWS CLI o un AWS SDK para API Gateway para probar la invocación de un método. Para hacerlo a través de la AWS CLI, consulte test-invoke-method.

Uso de un SDK de Java generado por API Gateway para una API REST

En esta sección se describen los pasos para utilizar un SDK de Java generado por API Gateway para una API REST mediante la API Calculadora simple como ejemplo. Antes de continuar, debe completar los pasos de Generación de SDK para las API de REST en API Gateway.

Para instalar y utilizar un SDK de Java generado por API Gateway

1. Extraiga el contenido del archivo .zip generado por API Gateway que ha descargado anteriormente.
2. Descargue e instale Apache Maven (versión 3.5 o posterior).
3. Descargar e instalar JDK 8.
4. Establezca la variable de entorno `JAVA_HOME`.
5. Vaya a la carpeta SDK descomprimida donde se encuentra el archivo pom.xml. Esta carpeta es `generated-code` de forma predeterminada. Ejecute el comando mvn install para instalar los archivos de artefactos compilados en el repositorio de Maven local. Se creará una carpeta `target` con la biblioteca del SDK compilada.

6. Escriba el comando siguiente en un directorio vacío para crear un stub del proyecto cliente que llame a la API utilizando la biblioteca de SDK instalada.

```
mvn -B archetype:generate \
    -DarchetypeGroupdId=org.apache.maven.archetypes \
    -DgroupId=examples.aws.apig.simpleCalc.sdk.app \
    -DartifactId=SimpleCalc-sdkClient
```

> **Note**
>
> El separador \ del comando anterior se ha incluido para facilitar la lectura. El comando completo debe estar en una sola línea sin el separador.

Este comando crea un stub de aplicación. El stub de aplicación contiene un archivo `pom.xml` y una carpeta `src` en el directorio raíz del proyecto (*`SimpleCalc-sdkClient`* en el comando anterior). Inicialmente, hay dos archivos de código fuente: `src/main/java/`*`{package-path}`*`/App.java` y `src/test/java/`*`{package-path}`*`/AppTest.java`. En este ejemplo, *`{package-path}`* es `examples/aws/apig/simpleCalc/sdk/app`. Esta ruta del paquete se obtiene del valor de `DarchetypeGroupdId`. Puede utilizar el archivo `App.java` como una plantilla para la aplicación cliente y puede añadir otros archivos en la misma carpeta si es necesario. Puede utilizar el archivo `AppTest.java` como una plantilla de pruebas unitarias para su aplicación y puede añadir otros archivos de código de prueba en la misma carpeta de pruebas si es necesario.

7. Actualice las dependencias del paquete en el archivo `pom.xml` generado de la forma siguiente, sustituyendo las propiedades `groupId`, `artifactId`, `version` y `name`, si es necesario:

```
<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**
>
> Cuando una versión más reciente de un artefacto dependiente de `aws-java-sdk-core` no es compatible con la versión especificada anteriormente (`1.11.94`), debe actualizar la etiqueta `<version>` a la nueva versión.

8. A continuación, mostramos cómo llamar a la API mediante el SDK llamando a los métodos `getABOp(GetABOpRequest req)`, `getApiRoot(GetApiRootRequest req)` y `postApiRoot(PostApiRootRequest req)` del SDK. Estos métodos se corresponden con los métodos `GET /{a}/{b}/{op}`, `GET /?a={x}&b={y}&op={operator}` y `POST /`, con una carga de solicitudes de API `{"a": x, "b": y, "op": "operator"}`, respectivamente.

   Actualice el archivo `App.java` como se indica a continuación:

```
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());


    }
}
```

En el ejemplo anterior, los ajustes de configuración que se utilizan para crear instancias del cliente SDK tienen una finalidad ilustrativa y no constituyen necesariamente prácticas recomendadas. Además, la llamada a `sdkClient.shutdown()` es opcional, especialmente si necesita un control preciso sobre el momento en que se van a liberar los recursos.

Hemos visto los principales patrones para llamar a una API utilizando un SDK de Java. Puede ampliar las instrucciones para llamar a otros métodos de API.

Uso de un SDK de Android generado por API Gateway para una API REST

En esta sección, se describen los pasos para utilizar un SDK de Android generado por API Gateway para una API REST. Antes de continuar, debe haber completado los pasos de Generación de SDK para las API de REST en API Gateway.

> **Note**
>
> El SDK generado no es compatible con Android 4.4 y versiones anteriores. Para obtener más información, consulte the section called “Notas importantes”.

Para instalar y utilizar un SDK de Android generado por API Gateway

1. Extraiga el contenido del archivo .zip generado por API Gateway que ha descargado anteriormente.

2. Descargue e instale Apache Maven (preferiblemente la versión 3.x).
3. Descargar e instalar JDK 8.
4. Establezca la variable de entorno `JAVA_HOME`.
5. Ejecute el comando mvn install para instalar los archivos de artefactos compilados en el repositorio de Maven local. Se creará una carpeta `target` con la biblioteca del SDK compilada.
6. Copie el archivo de SDK (cuyo nombre se obtiene de los elementos Artifact Id (ID de artefacto) y Artifact Version (Versión de artefacto) que especificó cuando generó el SDK, p. ej., `simple-calcsdk-1.0.0.jar`) desde la carpeta `target`, junto con todas las demás bibliotecas de la carpeta `target/lib` en la carpeta `lib` del proyecto.

   Si utiliza Android Studio, cree una carpeta `libs` en el módulo de la aplicación cliente y copie el archivo .jar necesario en esta carpeta. Compruebe que la sección de dependencias del archivo gradle del módulo contiene lo siguiente.

   ```
       compile fileTree(include: ['*.jar'], dir: 'libs')
       compile fileTree(include: ['*.jar'], dir: 'app/libs')
   ```

   Asegúrese de que no haya archivos .jar declarados duplicados.
7. Utilice la clase `ApiClientFactory` para inicializar el SDK generado por API Gateway. Por ejemplo:

   ```
   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. Para utilizar un proveedor de credenciales de Amazon Cognito para autorizar las llamadas a la API, utilice la clase `ApiClientFactory` para pasar un conjunto de credenciales de AWS mediante el SDK generado por API Gateway, tal y como se muestra en el siguiente ejemplo.

```
// 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. Para definir una clave de API mediante el SDK generado por API Gateway, utilice un código similar al siguiente.

```
ApiClientFactory factory = new ApiClientFactory()
  .apiKey("YOUR_API_KEY");
```

Uso de un SDK de JavaScript generado por API Gateway para una API REST

En el siguiente procedimiento se muestra cómo crear un SDK de JavaScript generado por API Gateway.

> **Note**
>
> En estas instrucciones se presupone que ya ha completado las instrucciones de Generación de SDK para las API de REST en API Gateway.

> **Important**
>
> Si su API solo tiene métodos ANY definidos, el paquete SDK generado no contendrá un archivo `apigClient.js`, por lo que tendrá que definir los métodos ANY usted mismo.

Para instalar, iniciar e invocar un SDK de JavaScript generado por API Gateway para una API REST

1. Extraiga el contenido del archivo .zip generado por API Gateway que ha descargado anteriormente.
2. Habilite el uso compartido de recursos de origen cruzado (CORS) para todos los métodos a los que llamará el SDK generado por API Gateway. Para obtener instrucciones, consulte CORS para las API de REST en API Gateway.
3. En la página web, incluya referencias a los siguientes scripts.

```
<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. En el código, inicialice el SDK generado por API Gateway mediante un código similar al siguiente.

```
var apigClient = apigClientFactory.newClient();
```

Para inicializar el SDK generado por API Gateway con credenciales de AWS, utilice un código similar al siguiente. Si utiliza las credenciales de AWS, todas las solicitudes a la API se firmarán.

```
var apigClient = apigClientFactory.newClient({
  accessKey: 'ACCESS_KEY',
  secretKey: 'SECRET_KEY',
});
```

Para utilizar una clave de API con el SDK generado por API Gateway, pase la clave de API como parámetro al objeto `Factory` utilizando un código similar al siguiente. Si utiliza una clave de API, esta se especifica como parte del encabezado `x-api-key` y todas las solicitudes a la API se firmarán. Esto significa que debe configurar los encabezados Accept de CORS adecuados para cada solicitud.

```
var apigClient = apigClientFactory.newClient({
  apiKey: 'API_KEY'
});
```

5. Llame a los métodos de la API en API Gateway con un código similar al siguiente. Cada llamada devuelve una promesa con devoluciones de llamada de éxito y fracaso.

```
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.
    });
```

Aquí, *methodName* se crea a partir de la ruta y el verbo HTTP del recurso de la solicitud de método. Para la API SimpleCalc, los métodos de SDK para los métodos de la API de

```
1. GET /?a=...&b=...&op=...
2. POST /

   { "a": ..., "b": ..., "op": ...}
3. GET /{a}/{b}/{op}
```

los métodos del SDK correspondientes son los siguientes:

```
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
```

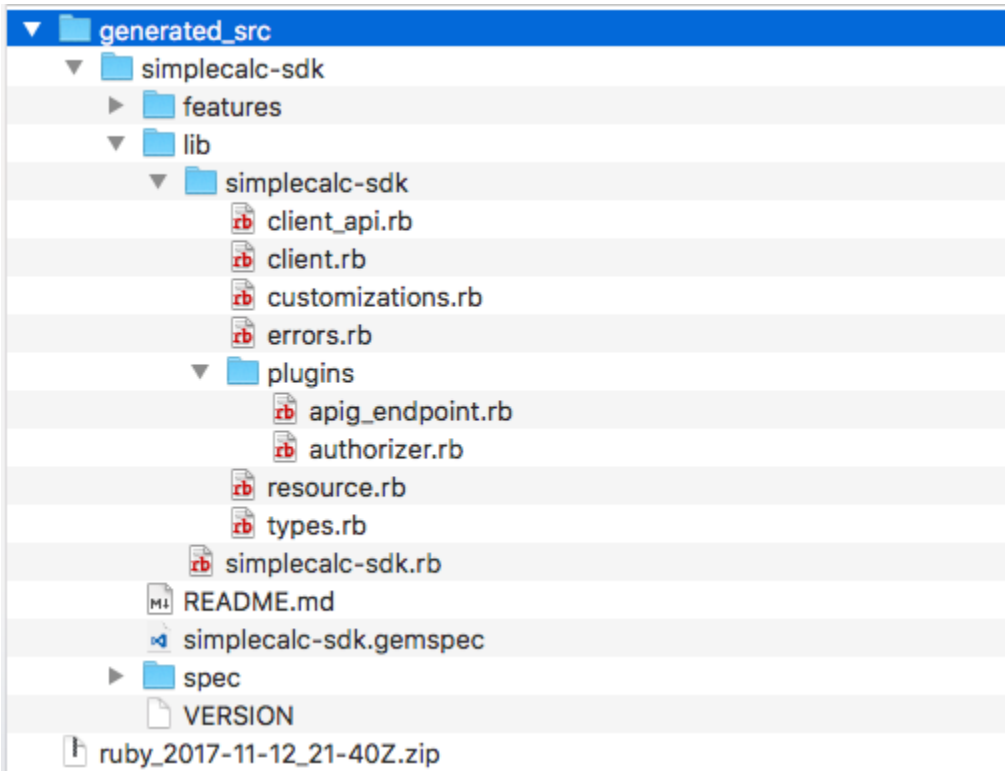
Uso de un SDK de Ruby generado por API Gateway para una API REST

En el siguiente procedimiento se muestra cómo crear un SDK de Ruby generado por API Gateway.

> **Note**
>
> En estas instrucciones, se presupone que ya se ha completado el procedimiento descrito en Generación de SDK para las API de REST en API Gateway.

Para instalar, instanciar e invocar un SDK de Ruby generado por API Gateway para una API REST

1. Descomprima el archivo descargado del SDK de Ruby. El código fuente del SDK generado es el siguiente.



2. Cree una gema de Ruby a partir del código fuente del SDK generado. Para ello, utilice los siguientes comandos shell en una ventana del terminal:

```
# change to /simplecalc-sdk directory
cd simplecalc-sdk

# build the generated gem
gem build simplecalc-sdk.gemspec
```

   Una vez realizada esta operación, simplecalc-sdk-1.0.0.gem pasa a estar disponible.

3. Instale la gema:

```
gem install simplecalc-sdk-1.0.0.gem
```

4. Cree una aplicación cliente. Instancie e inicialice el cliente del SDK de Ruby en la aplicación:

```
require 'simplecalc-sdk'
client = SimpleCalc::Client.new(
    http_wire_trace: true,
    retry_limit: 5,
    http_read_timeout: 50
```

```
)
```

Si la API tiene una autorización de tipo `AWS_IAM`, puede incluir las credenciales de AWS del intermediario suministrando `accessKey` y `secretKey` durante la inicialización:

```
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. Realice llamadas a la API a través del SDK de la aplicación.

> **Tip**
>
> Si no está familiarizado con las convenciones de llamada a métodos del SDK, puede consultar el archivo `client.rb` de la carpeta `lib` del SDK generado. La carpeta contiene documentación sobre las llamadas a cada uno de los métodos de API compatibles.

Para reconocer las operaciones admitidas:

```
# to show supported operations:
puts client.operation_names
```

Esto genera la siguiente llamada, que corresponde a los métodos de la API `GET /?a={.}&b={.}&op={.}`, `GET /{a}/{b}/{op}` y `POST /`, junto con una carga en el formato `{a:"…", b:"…", op:"…"}`, respectivamente:

```
[:get_api_root, :get_ab_op, :post_api_root]
```

Para invocar el método `GET /?a=1&b=2&op=+` de la API, llame al siguiente método del SDK de Ruby:

```
resp = client.get_api_root({a:"1", b:"2", op:"+"})
```

Para invocar el método `POST /` de la API con una carga `{a: "1", b: "2", "op": "+"}`, llame al siguiente método del SDK de Ruby:

```
resp = client.post_api_root(input: {a:"1", b:"2", op:"+"})
```

Para invocar al método `GET /1/2/+` de la API, llame al siguiente método del SDK de Ruby:

```
resp = client.get_ab_op({a:"1", b:"2", op:"+"})
```

Las llamadas al método del SDK que se realizan correctamente devuelven la siguiente respuesta:

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

Uso de un SDK de iOS generado por API Gateway para una API REST en Objective-C o Swift

En este tutorial, le mostraremos cómo utilizar un SDK de iOS generado por API Gateway para una API REST en una aplicación Objective-C o Swift para llamar a la API subyacente. Usaremos la API SimpleCalc como ejemplo para ilustrar los siguientes temas:

- Cómo instalar los componentes del SDK para móviles de AWS necesarios en el proyecto Xcode
- Cómo crear el objeto de cliente API antes de llamar a los métodos de la API

- Cómo llamar a los métodos de la API a través de los métodos del SDK correspondientes en el objeto del cliente API
- Cómo preparar una entrada de método y analizar el resultado utilizando las clases del modelo correspondientes del SDK

Temas

- Usar el SDK de iOS generado (Objective-C) para llamar a una API
- Usar un SDK de iOS generado (Swift) para llamar a la API

Usar el SDK de iOS generado (Objective-C) para llamar a una API

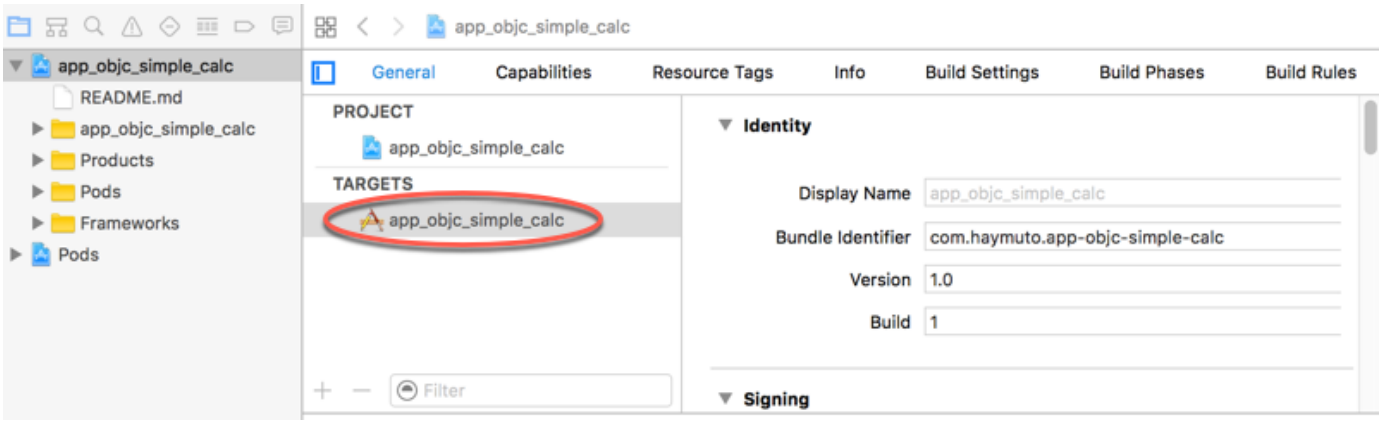
Antes de comenzar el siguiente procedimiento, debe completar los pasos de Generación de SDK para las API de REST en API Gateway para iOS en Objective-C y descargar el archivo .zip del SDK generado.

Instalación del SDK para móviles de AWS y un SDK de iOS generado por API Gateway en un proyecto Objective-C

En el procedimiento siguiente se describe cómo instalar el SDK.

Para instalar y utilizar un SDK de iOS generado por API Gateway en Objective-C

1. Extraiga el contenido del archivo .zip generado por API Gateway que ha descargado anteriormente. Con la API SimpleCalc, puede cambiar el nombre de la carpeta del SDK descomprimido a algo similar a **`sdk_objc_simple_calc`**. En esta carpeta del SDK hay un archivo `README.md` y un archivo `Podfile`. El archivo `README.md` contiene las instrucciones para instalar y utilizar el SDK. Este tutorial proporciona información detallada sobre estas instrucciones. La instalación usa CocoaPods para importar las bibliotecas de API Gateway necesarias y otros componentes dependientes del SDK para móviles de AWS. Debe actualizar el archivo `Podfile` para importar los SDK en el proyecto Xcode de su aplicación. La carpeta del SDK descomprimida contiene también una carpeta `generated-src` que incluye el código fuente del SDK generado de la API.
2. Inicie Xcode y cree un nuevo proyecto iOS Objective-C. Anote el destino del proyecto. Lo necesitará para definirlo en el archivo `Podfile`.



3. Para importar AWS Mobile SDK for iOS en el proyecto Xcode mediante CocoaPods, haga lo siguiente:

   a. Instale CocoaPods ejecutando el siguiente comando en una ventana del terminal:

   ```
   sudo gem install cocoapods
   pod setup
   ```

   b. Copie el archivo `Podfile` de la carpeta del SDK extraído en el mismo directorio que contiene el archivo del proyecto Xcode. Sustituya el siguiente bloque:

   ```
   target '<YourXcodeTarget>' do
       pod 'AWSAPIGateway', '~> 2.4.7'
   end
   ```

   por el nombre de destino de su proyecto:

   ```
   target 'app_objc_simple_calc' do
       pod 'AWSAPIGateway', '~> 2.4.7'
   end
   ```

   Si su proyecto Xcode ya contiene un archivo denominado `Podfile`, añada la siguiente línea de código:

   ```
   pod 'AWSAPIGateway', '~> 2.4.7'
   ```

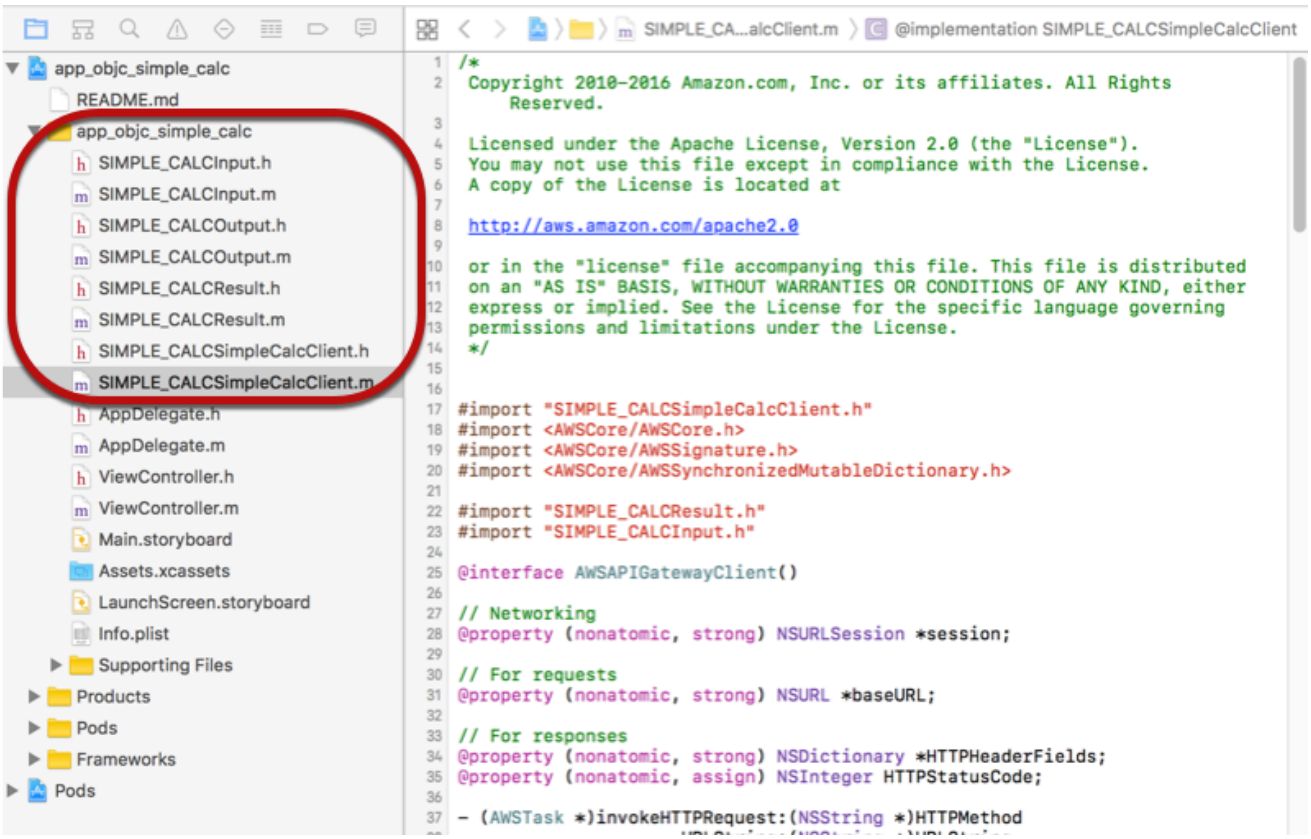
   c. Abra una ventana del terminal y ejecute el siguiente comando:

   ```
   pod install
   ```

Se instalará el componente de API Gateway y otros componentes dependientes del SDK para móviles de AWS.

d. Cierre el proyecto Xcode y, a continuación, abra el archivo `.xcworkspace` para volver a iniciar Xcode.

e. Añada todos los archivos `.h` y `.m` del directorio `generated-src` del SDK extraído al proyecto Xcode.



Para importar el AWS Mobile SDK for iOS Objective-C en su proyecto descargando de forma explícita el SDK para móviles de AWS o mediante Carthage, siga las instrucciones del archivo README.md. Asegúrese de utilizar únicamente una de estas opciones para importar el SDK para móviles de AWS.

Llamar a métodos de la API mediante el SDK de iOS generado por API Gateway en un proyecto Objective-C

Cuando genera el SDK con el prefijo `SIMPLE_CALC` para esta API SimpleCalc con dos modelos para la entrada (`Input`) y la salida (`Result`) de los métodos, en el SDK, la clase de cliente

de API resultante se convierte en `SIMPLE_CALCSimpleCalcClient` y las clases de datos correspondientes son `SIMPLE_CALCInput` y `SIMPLE_CALCResult`, respectivamente. Las solicitudes y respuestas de la API se asignan a los métodos del SDK de la siguiente manera:

- La solicitud de API

  ```
  GET /?a=...&b=...&op=...
  ```

  se convierte en el método del SDK

  ```
  (AWSTask *)rootGet:(NSString *)op a:(NSString *)a b:(NSString *)b
  ```

  La propiedad `AWSTask.result` es del tipo `SIMPLE_CALCResult` si el modelo `Result` se añadió a la respuesta del método. De lo contrario, la propiedad es del tipo `NSDictionary`.

- Esta solicitud de API

  ```
  POST /

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

  se convierte en el método del SDK

  ```
  (AWSTask *)rootPost:(SIMPLE_CALCInput *)body
  ```

- La solicitud de API

  ```
  GET /{a}/{b}/{op}
  ```

  se convierte en el método del SDK

  ```
  (AWSTask *)aBOpGet:(NSString *)a b:(NSString *)b op:(NSString *)op
  ```

El siguiente procedimiento describe cómo llamar a los métodos de la API en el código fuente de la aplicación Objective-C, por ejemplo, como parte del delegado `viewDidLoad` en un archivo `ViewController.m`.

Para llamar a la API a través del SDK de iOS generado por API Gateway

1. Importe el archivo de encabezados de clase del cliente API para que la clase del cliente API se pueda llamar en la aplicación:

   ```
   #import "SIMPLE_CALCSimpleCalc.h"
   ```

   La instrucción `#import` también importa `SIMPLE_CALCInput.h` y `SIMPLE_CALCResult.h` para las dos clases de modelo.

2. Cree una instancia de la clase del cliente de API:

   ```
   SIMPLE_CALCSimpleCalcClient *apiInstance = [SIMPLE_CALCSimpleCalcClient
    defaultClient];
   ```

   Para utilizar Amazon Cognito con la API, establezca la propiedad `defaultServiceConfiguration` en el objeto `AWSServiceManager` predeterminado, tal y como se muestra a continuación, antes de llamar al método `defaultClient` para crear el objeto del cliente API (mostrado en el ejemplo anterior):

   ```
   AWSCognitoCredentialsProvider *creds = [[AWSCognitoCredentialsProvider alloc]
    initWithRegionType:AWSRegionUSEast1 identityPoolId:your_cognito_pool_id];
   AWSServiceConfiguration *configuration = [[AWSServiceConfiguration alloc]
    initWithRegion:AWSRegionUSEast1 credentialsProvider:creds];
   AWSServiceManager.defaultServiceManager.defaultServiceConfiguration =
    configuration;
   ```

3. Llame al método `GET /?a=1&b=2&op=+` para ejecutar 1+2:

   ```
   [[apiInstance rootGet: @"+" a:@"1" b:@"2"] continueWithBlock:^id _Nullable(AWSTask
    * _Nonnull task) {
       _textField1.text = [self handleApiResponse:task];
       return nil;
   }];
   ```

donde la función auxiliar `handleApiResponse:task` formatea el resultado como una cadena que se muestra en un campo de texto (`_textField1`).

```
- (NSString *)handleApiResponse:(AWSTask *)task {
    if (task.error != nil) {
        return [NSString stringWithFormat: @"Error: %@", task.error.description];
    } else if (task.result != nil && [task.result isKindOfClass:[SIMPLE_CALCResult
 class]]) {
        return [NSString stringWithFormat:@"%@ %@ %@ = %@\n",task.result.input.a,
 task.result.input.op, task.result.input.b, task.result.output.c];
    }
    return nil;
}
```

El resultado que se muestra es `1 + 2 = 3`.

4. Llame al método `POST /` con una carga para ejecutar `1-2`:

```
SIMPLE_CALCInput *input = [[SIMPLE_CALCInput alloc] init];
    input.a = [NSNumber numberWithInt:1];
    input.b = [NSNumber numberWithInt:2];
    input.op = @"-";
    [[apiInstance rootPost:input] continueWithBlock:^id _Nullable(AWSTask *
 _Nonnull task) {
        _textField2.text = [self handleApiResponse:task];
        return nil;
    }];
```

El resultado que se muestra es `1 - 2 = -1`.

5. Llame al método `GET /{a}/{b}/{op}` para ejecutar `1/2`:

```
[[apiInstance aBOpGet:@"1" b:@"2" op:@"div"] continueWithBlock:^id
 _Nullable(AWSTask * _Nonnull task) {
    _textField3.text = [self handleApiResponse:task];
    return nil;
}];
```

El resultado que se muestra es `1 div 2 = 0.5`. Aquí se usa `div` en lugar de `/` porque la función de Lambda simple del backend no administra variables de ruta codificadas como URL.

Usar un SDK de iOS generado (Swift) para llamar a la API

Antes de comenzar el siguiente procedimiento, debe completar los pasos de Generación de SDK para las API de REST en API Gateway para iOS en Swift y descargar el archivo .zip del SDK generado.

Temas

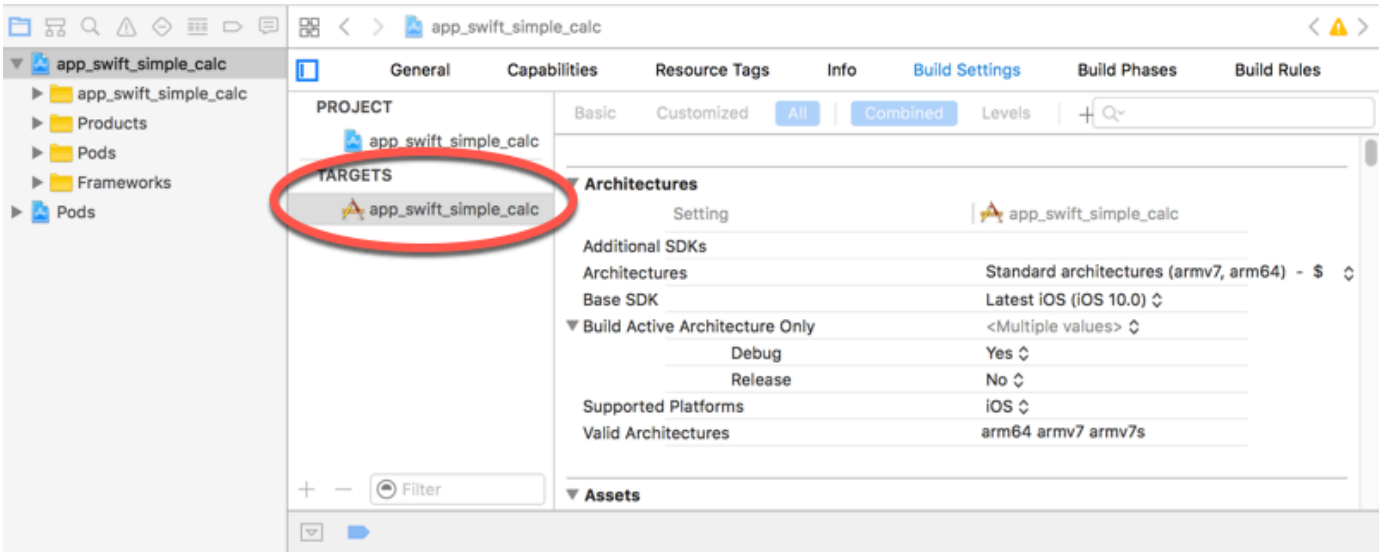
- Instalación del SDK para móviles de AWS y el SDK generado por API Gateway en un proyecto Swift
- Llamar a métodos de API a través del SDK de iOS generado por API Gateway en un proyecto Swift

Instalación del SDK para móviles de AWS y el SDK generado por API Gateway en un proyecto Swift

En el procedimiento siguiente se describe cómo instalar el SDK.

Para instalar y utilizar un SDK de iOS generado por API Gateway en Swift

1. Extraiga el contenido del archivo .zip generado por API Gateway que ha descargado anteriormente. Con la API SimpleCalc, puede cambiar el nombre de la carpeta del SDK descomprimido a algo similar a **`sdk_swift_simple_calc`**. En esta carpeta del SDK hay un archivo `README.md` y un archivo `Podfile`. El archivo `README.md` contiene las instrucciones para instalar y utilizar el SDK. Este tutorial proporciona información detallada sobre estas instrucciones. La instalación utiliza CocoaPods para importar los componentes del SDK para móviles de AWS necesarios. Debe actualizar el archivo `Podfile` para importar los SDK en el proyecto Xcode de su aplicación Swift. La carpeta del SDK descomprimida contiene también una carpeta `generated-src` que incluye el código fuente del SDK generado de la API.
2. Inicie Xcode y cree un nuevo proyecto de iOS Swift. Anote el destino del proyecto. Lo necesitará para definirlo en el archivo `Podfile`.



3. Para importar los componentes del SDK para móviles de AWS necesarios en el proyecto Xcode mediante CocoaPods, haga lo siguiente:

   a. Si no está instalado, instale CocoaPods ejecutando el siguiente comando en una ventana del terminal:

   ```
   sudo gem install cocoapods
   pod setup
   ```

   b. Copie el archivo `Podfile` de la carpeta del SDK extraído en el mismo directorio que contiene el archivo del proyecto Xcode. Sustituya el siguiente bloque:

   ```
   target '<YourXcodeTarget>' do
       pod 'AWSAPIGateway', '~> 2.4.7'
   end
   ```

   por el nombre de destino de su proyecto, tal y como se muestra a continuación:

   ```
   target 'app_swift_simple_calc' do
       pod 'AWSAPIGateway', '~> 2.4.7'
   end
   ```

   Si su proyecto Xcode ya contiene un archivo `Podfile` con el destino correcto, solo tiene que añadir la siguiente línea de código al bucle `do ... end`:

   ```
   pod 'AWSAPIGateway', '~> 2.4.7'
   ```

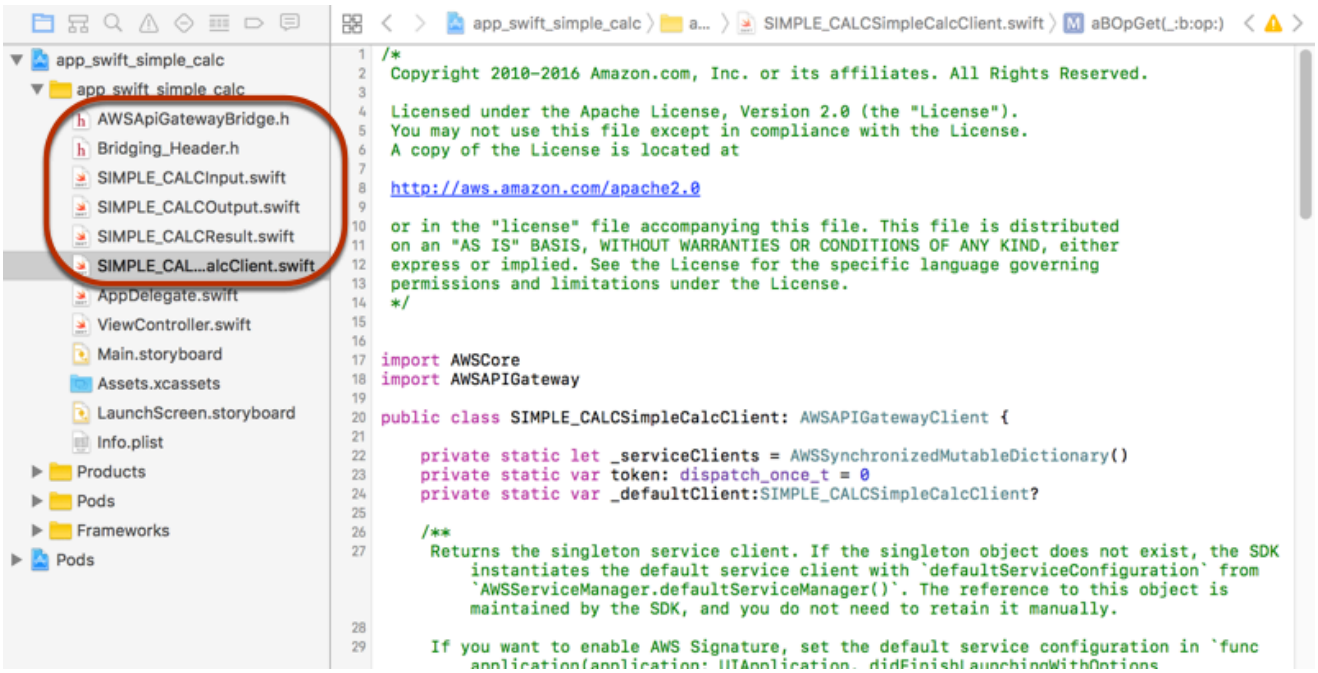
c. Abra una ventana del terminal y ejecute el siguiente comando en el directorio de la aplicación:

```
pod install
```

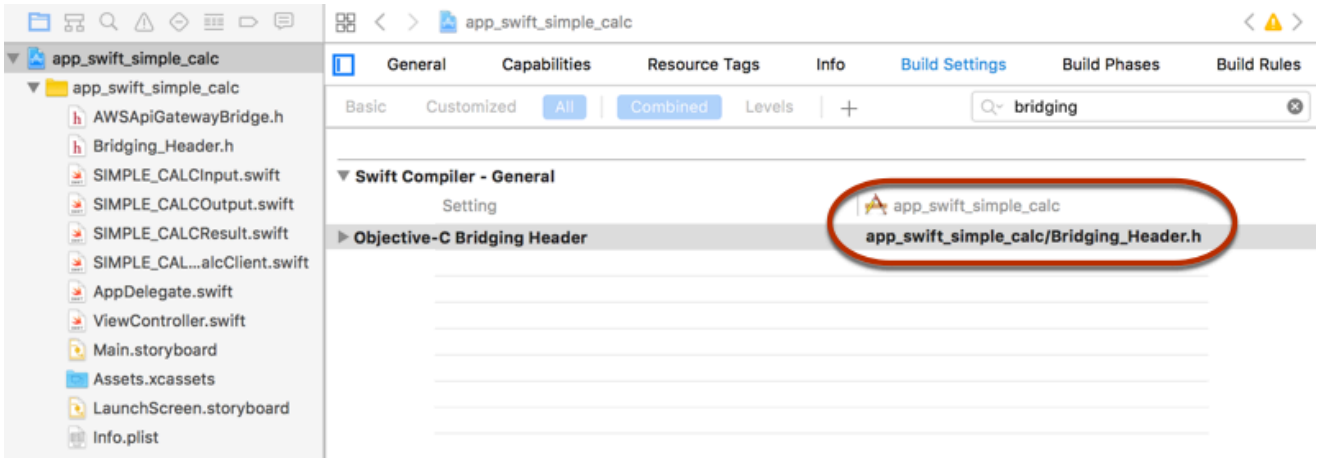
Se instalará el componente de API Gateway y todos los componentes dependientes del SDK para móviles de AWS en el proyecto de la aplicación.

d. Cierre el proyecto Xcode y, a continuación, abra el archivo `*.xcworkspace` para volver a iniciar Xcode.

e. Añada todos los archivos de encabezado del SDK (`.h`) y los archivos de código fuente de Swift (`.swift`) del directorio `generated-src` extraído a su proyecto Xcode.

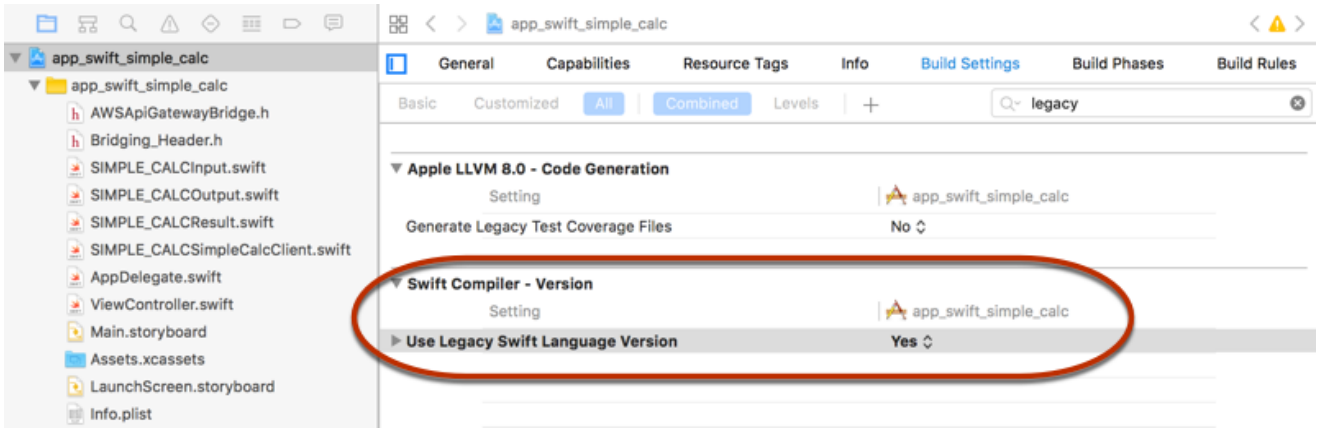


f. Para habilitar las llamadas a las bibliotecas de Objective-C del SDK para móviles de AWS de su proyecto de código Swift, defina la ruta del archivo `Bridging_Header.h` en la propiedad Objective-C Bridging Header (Encabezado Bridging de Objective-C) en la opción Swift Compiler - General (Compilador de Swift: general) de la configuración de su proyecto Xcode:



> **Tip**
>
> Puede escribir **`bridging`** en el cuadro de búsqueda de Xcode para encontrar la propiedad Objective-C Bridging Header (Encabezado Bridging de Objective-C).

g. Compile el proyecto Xcode para verificar que está configurado correctamente antes de continuar. Si su Xcode utiliza una versión más reciente de Swift que la admitida por el SDK para móviles de AWS, recibirá errores del compilador de Swift. En este caso, establezca la propiedad Use Legacy Swift Language Version (Usar versión del lenguaje Swift antigua) en Yes (Sí) en la opción Swift Compiler - Version (Compilador de Swift: versión):



Para importar el SDK para móviles para iOS de AWS en Swift en su proyecto descargando explícitamente el SDK para móviles de AWS o utilizando Carthage, siga las instrucciones del archivo `README.md` incluido con el paquete del SDK. Asegúrese de utilizar únicamente una de estas opciones para importar el SDK para móviles de AWS.

Llamar a métodos de API a través del SDK de iOS generado por API Gateway en un proyecto Swift

Cuando genera el SDK con el prefijo `SIMPLE_CALC` para esta API SimpleCalc con dos modelos para describir la entrada (`Input`) y la salida (`Result`) de las solicitudes y las respuestas de la API, en el SDK, la clase de cliente de API resultante se convierte en `SIMPLE_CALCSimpleCalcClient` y las clases de datos correspondientes son `SIMPLE_CALCInput` y `SIMPLE_CALCResult`, respectivamente. Las solicitudes y respuestas de la API se asignan a los métodos del SDK de la siguiente manera:

- La solicitud de API

  ```
  GET /?a=...&b=...&op=...
  ```

  se convierte en el método del SDK

  ```
  public func rootGet(op: String?, a: String?, b: String?) -> AWSTask
  ```

  La propiedad `AWSTask.result` es del tipo `SIMPLE_CALCResult` si el modelo `Result` se añadió a la respuesta del método. De lo contrario, es del tipo `NSDictionary`.

- Esta solicitud de API

  ```
  POST /

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

  se convierte en el método del SDK

  ```
  public func rootPost(body: SIMPLE_CALCInput) -> AWSTask
  ```

- La solicitud de API

  ```
  GET /{a}/{b}/{op}
  ```

  se convierte en el método del SDK

```
public func aBOpGet(a: String, b: String, op: String) -> AWSTask
```

El siguiente procedimiento describe cómo llamar a los métodos de la API en el código fuente de la aplicación Swift, por ejemplo, como parte del delegado `viewDidLoad()` en un archivo `ViewController.m`.

Para llamar a la API a través del SDK de iOS generado por API Gateway

1. Cree una instancia de la clase del cliente de API:

   ```
   let client = SIMPLE_CALCSimpleCalcClient.default()
   ```

   Para utilizar Amazon Cognito con la API, establezca una configuración del servicio de AWS predeterminada (que se muestra a continuación) antes de obtener el método `default` (mostrado anteriormente):

   ```
   let credentialsProvider =
    AWSCognitoCredentialsProvider(regionType: AWSRegionType.USEast1, identityPoolId:
    "my_pool_id")
   let configuration = AWSServiceConfiguration(region: AWSRegionType.USEast1,
    credentialsProvider: credentialsProvider)
   AWSServiceManager.defaultServiceManager().defaultServiceConfiguration =
    configuration
   ```

2. Llame al método `GET /?a=1&b=2&op=+` para ejecutar 1+2:

   ```
   client.rootGet("+", a: "1", b:"2").continueWithBlock {(task: AWSTask) -> AnyObject?
    in
       self.showResult(task)
       return nil
   }
   ```

   donde la función auxiliar `self.showResult(task)` imprime el resultado o el error en la consola; por ejemplo:

   ```
   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)")
        }
    }
}
```

En una aplicación de producción, puede mostrar el resultado o el error en un campo de texto. El resultado que se muestra es `1 + 2 = 3`.

3. Llame al método `POST /` con una carga para ejecutar `1-2`:

```
let body = SIMPLE_CALCInput()
body.a=1
body.b=2
body.op="-"
client.rootPost(body).continueWithBlock {(task: AWSTask) -> AnyObject? in
    self.showResult(task)
    return nil
}
```

El resultado que se muestra es `1 - 2 = -1`.

4. Llame al método `GET /{a}/{b}/{op}` para ejecutar `1/2`:

```
client.aBOpGet("1", b:"2", op:"div").continueWithBlock {(task: AWSTask) ->
 AnyObject? in
    self.showResult(task)
    return nil
}
```

El resultado que se muestra es `1 div 2 = 0.5`. Aquí se usa `div` en lugar de `/` porque la función de Lambda simple del backend no administra variables de ruta codificadas como URL.

Desarrollo de API de REST mediante OpenAPI en API Gateway

Puede utilizar API Gateway para importar una API REST desde un archivo de definición externo a API Gateway. Actualmente, API Gateway es compatible con los archivos de definición OpenAPI v2.0 y OpenAPI v3.0, con las excepciones que se indican en Notas importantes de Amazon API Gateway para las API REST. Puede actualizar una API sobrescribiéndola con una nueva definición o puede combinar una definición con una API existente. Las opciones se especifican utilizando un parámetro de consulta `mode` en la URL de solicitud.

Para ver un tutorial sobre el uso de la característica Importar API desde la consola de API Gateway, consulte Tutorial: Crear una API de REST importando un ejemplo.

Temas

- Importación de una API optimizada para bordes en API Gateway
- Importación de una API regional en API Gateway
- Importación de un archivo de OpenAPI para actualizar una definición de la API existente
- Establezca la propiedad basePath de OpenAPI
- AWSVariables de para la importación de OpenAPI
- Errores y advertencias al importar la API a API Gateway
- Exportación de una API REST desde API Gateway

Importación de una API optimizada para bordes en API Gateway

Puede importar un archivo de definición de OpenAPI de la API para crear una nueva API optimizada para bordes especificando el tipo de punto de conexión `EDGE` como entrada adicional, además del archivo de OpenAPI, en la operación de importación. Puede hacerlo mediante la consola de API Gateway, la AWS CLI o un SDK de AWS.

Para ver un tutorial sobre el uso de la característica Importar API desde la consola de API Gateway, consulte Tutorial: Crear una API de REST importando un ejemplo.

Temas

- Importación de una API optimizada para bordes mediante la consola de API Gateway
- Importación de una API optimizada para bordes a través de la AWS CLI

Importación de una API optimizada para bordes mediante la consola de API Gateway

Si desea importar una API optimizada para bordes a través de la consola de API Gateway, haga lo siguiente:

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Seleccione Crear API.
3. En API de REST, elija Importar.
4. Copie una definición de OpenAPI de la API y péguela en el editor de código o elija Elegir archivo para cargar un archivo de OpenAPI de una unidad local.
5. En Tipo de punto de conexión de la API, seleccione Optimizado para bordes.
6. Elija Crear API para empezar a importar las definiciones de OpenAPI.

Importación de una API optimizada para bordes a través de la AWS CLI

El siguiente comando import-rest-api permite importar una API desde un archivo de definición de OpenAPI para crear una API optimizada para bordes:

```
aws apigateway import-rest-api \
    --fail-on-warnings \
    --body 'file://path/to/API_OpenAPI_template.json'
```

También puede establecer explícitamente el parámetro de la cadena de consulta `endpointConfigurationTypes` en `EDGE`:

```
aws apigateway import-rest-api \
    --parameters endpointConfigurationTypes=EDGE \
    --fail-on-warnings \
    --body 'file://path/to/API_OpenAPI_template.json'
```

Importación de una API regional en API Gateway

Al importar una API, puede elegir la configuración del punto de conexión regional de la API. Puede utilizar la consola de API Gateway, la AWS CLI o un SDK de AWS.

Al exportar una API, la configuración del punto de conexión de la API no se incluye en las definiciones de API exportadas.

Para ver un tutorial sobre el uso de la característica Importar API desde la consola de API Gateway, consulte Tutorial: Crear una API de REST importando un ejemplo.

Temas

- Importar una API regional mediante la consola de API Gateway
- Importación de una API regional con la AWS CLI

Importar una API regional mediante la consola de API Gateway

Para importar una API de un punto de conexión regional mediante la consola de API Gateway, haga lo siguiente:

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Seleccione Crear API.
3. En API de REST, elija Importar.
4. Copie una definición de OpenAPI de la API y péguela en el editor de código o elija Elegir archivo para cargar un archivo de OpenAPI de una unidad local.
5. En Tipo de punto de conexión de la API, seleccione Regional.
6. Elija Crear API para empezar a importar las definiciones de OpenAPI.

Importación de una API regional con la AWS CLI

El siguiente comando import-rest-api importa un archivo de definición de OpenAPI y establece el tipo de punto de conexión en Regional:

```
aws apigateway import-rest-api \
    --parameters endpointConfigurationTypes=REGIONAL \
    --fail-on-warnings \
    --body 'file://path/to/API_OpenAPI_template.json'
```

Importación de un archivo de OpenAPI para actualizar una definición de la API existente

Puede importar las definiciones de la API únicamente para actualizar una API existente, sin cambiar la configuración del punto de enlace, así como las etapas y las variables de etapa o referencias a las claves de API.

La operación de importar para actualizar puede producirse de dos modos: combinación o sobreescritura.

Cuando una API (A) se combina con otra (B), la API resultante conserva las definiciones de A y B si las dos API no comparten definiciones contradictorias. Si surge un conflicto, las definiciones de método de la combinación de la API (A) invalida las definiciones de método correspondientes de la API combinada (B). Suponga, por ejemplo, que B ha declarado los siguientes métodos para devolver las respuestas 200 y 206:

```
GET /a
POST /a
```

y A declara el siguiente método para devolver las respuestas 200 y 400:

```
GET /a
```

Cuando A se combina con B, la API resultante presenta los siguientes métodos:

```
GET /a
```

que devuelve las respuestas 200 y 400, y

```
POST /a
```

que devuelve las respuestas 200 y 206.

Combinar una API es útil cuando ha descompuesto sus definiciones de API externas en varias partes más pequeñas y solo desea aplicar los cambios de una de esas partes a la vez. Esto podría ocurrir, por ejemplo, si varios equipos son responsables de diferentes partes de una API y han realizado cambios disponibles en diferentes partes. En este modo, los elementos de la API existente que no están específicamente definidos en la definición importada se dejan tal como están.

Cuando una API (A) sobrescribe otra API (B), la API resultante adopta las definiciones de la API sobrescrita (A). Sobrescribir una API es útil cuando una definición de API externa contiene la definición completa de una API. En este modo, los elementos de una API existente que no están específicamente definidos en la definición importada se eliminan.

> **Important**
>
> Cuando usa `mode=overwrite`, se reemplaza toda la definición de la API. Las configuraciones de la API que no se incluyen explícitamente en el cuerpo de importación se restablecen a sus valores predeterminados. Esto incluye ajustes como las políticas de seguridad, las configuraciones de autorización y otras propiedades por API.
> Para conservar las configuraciones existentes, utilice `mode=merge`, que preserva las configuraciones existentes que no estén especificadas en el cuerpo de importación. Si usa `mode=overwrite`, asegúrese de que la definición de OpenAPI incluya todas las configuraciones de API deseadas, como las extensiones de API Gateway.

Para combinar una API, envíe una solicitud PUT a `https://apigateway.<region>.amazonaws.com/restapis/<restapi_id>?mode=merge`. El valor del parámetro de ruta `restapi_id` especifica la API con la que debe combinarse la definición de API proporcionada.

El siguiente fragmento de código muestra un ejemplo de la solicitud PUT para combinar una definición de la API de OpenAPI en JSON, como la carga, con la API ya especificada en API Gateway.

```
PUT /restapis/<restapi_id>?mode=merge
Host:apigateway.<region>.amazonaws.com
Content-Type: application/json
Content-Length: ...

An OpenAPI API definition in JSON
```

La operación de actualización por combinación toma dos definiciones de API completas y las combina. Para un cambio pequeño e incremental, puede usar la operación resource update.

Para sobrescribir una API, envíe una solicitud PUT a `https://apigateway.<region>.amazonaws.com/restapis/<restapi_id>?mode=overwrite`. El parámetro de ruta `restapi_id` especifica la API que se sobrescribirá con las definiciones de API proporcionadas.

El siguiente fragmento de código muestra un ejemplo de una solicitud de sobrescritura con la carga de una definición de OpenAPI con formato JSON:

```
PUT /restapis/<restapi_id>?mode=overwrite
Host:apigateway.<region>.amazonaws.com
Content-Type: application/json
Content-Length: ...

An OpenAPI API definition in JSON
```

Cuando no se especifica el parámetro de consulta `mode`, se supone que se trata de una combinación.

> **Note**
>
> Las operaciones PUT son idempotentes, pero no atómicas. Eso significa que si se produce un error del sistema durante el procesamiento, la API puede acabar en mal estado. Sin embargo, si se repite la operación de forma exitosa, la API acaba en el mismo estado final que tendría si la primera operación se hubiera realizado con éxito.

Establezca la propiedad **`basePath`** de OpenAPI

En OpenAPI 2.0, puede utilizar la propiedad `basePath` para proporcionar una o varias partes de la ruta que preceden a cada una de las rutas definidas en la propiedad `paths`. Como API Gateway tiene varias maneras de expresar la ruta de un recurso, la característica Importar API ofrece las siguientes opciones para interpretar la propiedad `basePath` durante una importación: "ignore", "prepend" y "split".

En OpenAPI 3.0, `basePath` ya no es una propiedad de nivel superior. En su lugar, API Gateway utiliza una variable de servidor como una convención. La característica Import API proporciona las mismas opciones para interpretar la ruta base durante la importación. La ruta base se identifica como se indica a continuación:

- Si la API no contiene ninguna variable `basePath`, la característica Import API comprueba la cadena `server.url` para ver si contiene una ruta más allá `"/"`. Si es así, esa ruta se utiliza como la base de la ruta.

- Si la API contiene una única variable `basePath`, la característica Import API la utiliza como la ruta base, incluso si no se hace referencia en el `server.url`.
- Si la API contiene varias variables `basePath`, la característica Import API utiliza únicamente la primera como la ruta base.

Ignore

Si el archivo de OpenAPI tiene un valor `basePath` de `/a/b/c` y la propiedad `paths` contiene `/e` y `/f`, la siguiente solicitud `POST` o `PUT`:

```
POST /restapis?mode=import&basepath=ignore
```

```
PUT /restapis/api_id?basepath=ignore
```

produce los siguientes recursos en la API:

- `/`
- `/e`
- `/f`

El efecto consiste en tratar `basePath` como si no estuviera presente, y todos los recursos de la API declarados se sirven en relación con el host. Esto puede utilizarse, por ejemplo, cuando disponga de un nombre de dominio personalizado con una asignación de API que no incluya un valor Base Path ni Stage que haga referencia a la etapa de producción.

> **Note**
>
> API Gateway crea automáticamente un recurso raíz, aunque no se haya declarado explícitamente en el archivo de definición.

Cuando no se especifica, `basePath` toma `ignore` como valor predeterminado.

Anexar

Si el archivo de OpenAPI tiene un valor `basePath` de `/a/b/c` y la propiedad `paths` contiene `/e` y `/f`, la siguiente solicitud `POST` o `PUT`:

```
POST /restapis?mode=import&basepath=prepend
```

```
PUT /restapis/api_id?basepath=prepend
```

produce los siguientes recursos en la API:

- `/`
- `/a`
- `/a/b`
- `/a/b/c`
- `/a/b/c/e`
- `/a/b/c/f`

El efecto consiste en tratar `basePath` como si especificara recursos adicionales (sin métodos) y añadiera estos recursos al conjunto de recursos declarados. Esto puede utilizarse, por ejemplo, cuando diferentes equipos sean responsables de diferentes partes de una API y `basePath` pueda hacer referencia a la ubicación de ruta de la parte de la API de cada equipo.

> **Note**
>
> API Gateway crea automáticamente los recursos intermedios, aunque no se hayan declarado explícitamente en la definición.

Split

Si el archivo de OpenAPI tiene un valor `basePath` de `/a/b/c` y la propiedad `paths` contiene `/e` y `/f`, la siguiente solicitud `POST` o `PUT`:

```
POST /restapis?mode=import&basepath=split
```

```
PUT /restapis/api_id?basepath=split
```

produce los siguientes recursos en la API:

- `/`
- `/b`
- `/b/c`
- `/b/c/e`
- `/b/c/f`

El efecto consiste en tratar la parte de la ruta superior, `/a`, como el principio de la ruta de cada recurso y crear recursos adicionales (sin método) dentro de la propia API. Esto podría usarse, por ejemplo, cuando a es un nombre de etapa que desee exponer como parte de la API.

AWSVariables de para la importación de OpenAPI

Puede utilizar las siguientes variables de AWS en las definiciones de OpenAPI. API Gateway resuelve las variables cuando se importa la API. Para especificar una variable, utilice `${`*`variable-name`*`}`. En la siguiente tabla se describen las variables de AWS disponibles.

Nombre de variable	Descripción
`AWS::AccountId`	El ID de cuenta de AWS que importa la API. Por ejemplo: 123456789012.
`AWS::Partition`	La partición de AWS en la que se importa la API. Para las regiones estándar de AWS, la partición es `aws`.
`AWS::Region`	La región de AWS en la que se importa la API. Por ejemplo, `us-east-2` .

AWSEjemplo de variables de

En el ejemplo siguiente, se utilizan variables de AWS para especificar una función de AWS Lambda para una integración.

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
```

Errores y advertencias al importar la API a API Gateway

Al importar el archivo de definición externo a API Gateway, API Gateway puede generar advertencias y errores. En las siguientes secciones se describen los errores y las advertencias que pueden producirse durante la importación.

Errores durante la importación

Durante la importación, se pueden generar errores por problemas graves, como un documento de OpenAPI no válido. Los errores se devuelven como excepciones (por ejemplo, `BadRequestException`) en una respuesta incorrecta. Cuando se produce un error, la nueva definición de la API se descarta y no se realiza ningún cambio en la API existente.

Advertencias durante la importación

Durante la importación, se pueden generar advertencias por problemas menores, como la falta de una referencia del modelo. Si se produce una advertencia, la operación continuará si la expresión de consulta `failonwarnings=false` se añade a la URL de la solicitud. De lo contrario, las actualizaciones se revertirán. De forma predeterminada, `failonwarnings` está establecido en `false`. En estos casos, las advertencias se devuelven como un campo en el recurso RestApi resultante. De lo contrario, las advertencias se devuelven como un mensaje en la excepción.

Exportación de una API REST desde API Gateway

Una vez que haya creado y configurado una API REST en API Gateway mediante la consola de API Gateway o de otra forma, puede exportarla a un archivo de OpenAPI mediante la característica Exportar API de API Gateway, que forma parte del servicio de control de Amazon API Gateway. Para usar la API de exportación de API Gateway, debe firmar sus solicitudes de API. Para obtener más información sobre la firma de solicitudes, consulte Firma de solicitudes de AWS en la Guía del usuario de IAM. Dispone de opciones para incluir las extensiones de integración de API Gateway, así como las extensiones Postman, en el archivo de definición de OpenAPI exportado.

> **Note**
>
> Al exportar la API utilizando la AWS CLI, asegúrese de incluir el parámetro de la extensión tal y como se muestra en el siguiente ejemplo, para garantizar que se incluye la extensión `x-amazon-apigateway-request-validator`:
>
> ```
> aws apigateway get-export --parameters extensions='apigateway' --rest-api-id
>  abcdefg123 --stage-name dev --export-type swagger latestswagger2.json
> ```

No puede exportar una API si sus cargas no son del tipo `application/json`. Si lo intenta, obtendrá una respuesta de error en la que se indica que no se encuentran los modelos del cuerpo JSON.

Solicitud de exportación de una API de REST

Con la API Export, puede exportar una API de REST existente enviando una solicitud GET en la que se especifique la API que se va exportar como parte de las rutas URL. La URL de la solicitud debe tener el siguiente formato:

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
```

Puede asociar la cadena de consulta `extensions` para especificar si desea incluir extensiones de API Gateway (con el valor `integration`) o extensiones de Postman (con el valor `postman`).

Además, puede establecer el encabezado `Accept` en `application/json` o `application/yaml` para recibir la salida de la definición de la API en formato JSON o YAML, respectivamente.

Para obtener más información sobre cómo enviar solicitudes GET utilizando la función Exportar API de API Gateway, consulte GetExport.

> **Note**
>
> Si define modelos en la API, deben ser para el tipo de contenido de "application/json" para que API Gateway pueda exportar el modelo. De lo contrario, API Gateway produce una excepción con el mensaje de error "Only found non-JSON body models for ...".
> Los modelos deben contener propiedades o definirse como un tipo de JSONSchema determinado.

Descarga de la definición de OpenAPI de la API de REST en JSON

Para exportar y descargar una API de REST en definiciones de OpenAPI con formato JSON:

OpenAPI 3.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/oas30
Host: apigateway.<region>.amazonaws.com
Accept: application/json
```

OpenAPI 2.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/swagger
Host: apigateway.<region>.amazonaws.com
Accept: application/json
```

Aquí, *<region>* podría ser, por ejemplo, `us-east-1`. Para todas las regiones donde API Gateway está disponible, consulte Regiones y puntos de conexión.

Descarga de la definición de OpenAPI de la API de REST en YAML

Para exportar y descargar una API de REST en definiciones de OpenAPI con formato YAML:

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
```

Descarga de la definición de OpenAPI de la API de REST con extensiones Postman en JSON

Para exportar y descargar una API de REST en definiciones de OpenAPI con Postman en formato JSON:

OpenAPI 3.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/oas30?extensions=postman
Host: apigateway.<region>.amazonaws.com
Accept: application/json
```

OpenAPI 2.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/swagger?extensions=postman
Host: apigateway.<region>.amazonaws.com
Accept: application/json
```

Descarga de la definición de OpenAPI de la API REST con la integración de API Gateway en YAML

Para exportar y descargar una API REST en definiciones de OpenAPI con la integración de API Gateway en formato YAML:

OpenAPI 3.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/oas30?extensions=integrations
Host: apigateway.<region>.amazonaws.com
Accept: application/yaml
```

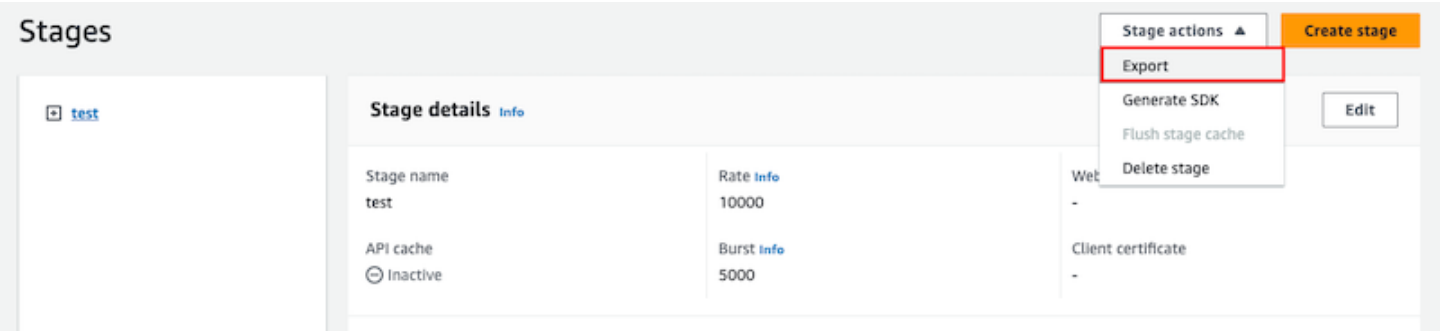
OpenAPI 2.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/swagger?
extensions=integrations
Host: apigateway.<region>.amazonaws.com
Accept: application/yaml
```

Exportar API REST con la consola de API Gateway

Después de implementar la API REST en una etapa, puede empezar a exportar la API en la etapa a un archivo de OpenAPI mediante la consola de API Gateway.

En el panel Etapas de la consola de API Gateway, elija Acciones de etapa, Exportar.



Especifique un Tipo de especificación de API, Formato y Extensiones para descargar la definición de OpenAPI de la API.

Publicación de las API de REST para que los clientes las invoquen

El hecho de crear y desarrollar simplemente una API de API Gateway no hace que automáticamente los usuarios puedan invocarla. Para que sea invocable, debe implementar su API en una etapa. Además, es posible que desee personalizar la URL que utilizarán los usuarios para acceder a su API. Puedes darle un dominio que sea coherente con su marca o que sea más memorable que la URL predeterminada de su API.

En esta sección, puede aprender a implementar su API y personalizar la URL que proporciona a los usuarios para acceder a ella.

> **Note**
>
> Para aumentar la seguridad de las API de API Gateway, el dominio `execute-api.{region}.amazonaws.com` se registra en la lista de sufijos públicos (PSL). Para mayor seguridad, recomendamos que utilice cookies con un prefijo `__Host-` en caso de que necesite configurar cookies confidenciales en el nombre de dominio predeterminado de las API de API Gateway. Esta práctica le ayudará a proteger su dominio de los intentos de falsificación de solicitudes entre sitios (CSRF). Para obtener más información, consulte la página de configuración de cookies en la red de desarrolladores de Mozilla.

Temas

- Implementación de las API de REST en API Gateway
- Nombre de dominio personalizado para las API de REST públicas en API Gateway

Implementación de las API de REST en API Gateway

Después de crear la API, debe implementarla para que los usuarios puedan llamarla.

Para implementar una API, debe crear una implementación de API y asociarla con una etapa. Una referencia lógica a un estado del ciclo de vida de la API (por ejemplo, `dev`, `prod`, `beta`, `v2`). Las etapas de API se identifican por un ID y un nombre de etapa de API. Se incluyen en la URL que se utiliza para invocar la API. Cada etapa es una referencia con nombre a una implementación de la API y está disponible para que las aplicaciones cliente la invoquen.

> **Important**
>
> Cada vez que actualice una API, debe volver a implementar la API en una etapa existente o en una nueva etapa. La actualización de una API incluye la modificación de rutas, métodos, integraciones, autorizadores, políticas de recursos y cualquier otra cosa que no sea la configuración de la etapa.

A medida que la API va evolucionando, se puede seguir implementando en diferentes etapas como versiones distintas. También puede implementar las actualizaciones de API como una implementación de lanzamiento canary. Esto permite a sus clientes API acceder, en el mismo escenario, a la versión de producción a través de la edición de producción, y a la versión actualizada a través de la edición canary.

Para llamar a una API implementada, el cliente envía una solicitud a una URL de la API. La dirección URL viene determinada por el protocolo de la API (HTTP(S) o (WSS)), el nombre de host, el nombre de la etapa y (para las API de REST) la ruta del recurso. El nombre de host y el nombre de etapa determinan la URL base de la API.

Por ejemplo, si se utiliza el nombre de dominio predeterminado de la API, la URL base de una API REST en una determinada etapa (*`{stageName}`*) tendrá el siguiente formato:

```
https://{restapi-id}.execute-api.{region}.amazonaws.com/{stageName}
```

Para simplificar la URL base predeterminada de una API para los usuarios, puede crear un nombre de dominio personalizado (por ejemplo, `api.example.com`) para sustituir el nombre de host predeterminado de la API. Para administrar varias API bajo el mismo nombre de domino, debe asignar una etapa de API a una ruta de acceso base.

Con el nombre de dominio personalizado *`{api.example.com}`* y la etapa de API mapeada a la ruta base (*`{basePath}`*) bajo el nombre de dominio personalizado, la URL base de una API de REST sería:

```
https://{api.example.com}/{basePath}
```

Para cada etapa, puede optimizar el rendimiento de la API ajustando los límites de solicitudes de nivel de cuenta predeterminados y habilitando el almacenamiento en caché de la API. También puede activar el registro de las llamadas a la API a CloudTrail o CloudWatch y puede seleccionar

un certificado de cliente para que el backend autentique las solicitudes de la API. Además, puede invalidar la configuración de nivel de etapa en los distintos métodos y definir variables de etapa para pasar contextos de entorno específicos de la etapa a la integración de la API en tiempo de ejecución.

Las etapas permiten realizar un exhaustivo control de versiones de la API. Por ejemplo, puede implementar una API en una etapa `test` y una etapa `prod`, y utilizar la etapa `test` como compilación de pruebas y la etapa `prod` como compilación estable. Una vez que las actualizaciones superan la prueba, puede promover la etapa `test` como etapa `prod`. La promoción puede hacerse mediante la nueva implementación de la API en la etapa `prod` o la actualización de un valor de variable de etapa del nombre de etapa de `test` a `prod`.

En esta sección vamos a hablar sobre cómo implementar una API con la consola de API Gateway o llamando a la API de REST de API Gateway. Para utilizar otras herramientas, consulte la documentación de la CLI de AWS o de un AWS SDK.

Temas

- Creación de una implementación de una API de REST en API Gateway
- Configuración de una etapa para una API de REST en API Gateway
- Configuración de una implementación de versión canario de API Gateway
- Actualizaciones de las API de REST que requieren reimplementación

Creación de una implementación de una API de REST en API Gateway

En API Gateway, la implementación de una API de REST se representa mediante un recurso Deployment. Es similar a un ejecutable de una API al cual representa un recurso RestAPI.

Para que el cliente pueda llamar a la API, debe crear una implementación y asociarle una etapa. Las etapas se representan mediante recursos Stage. Representa una instantánea de la API que incluye métodos, integraciones, modelos, plantillas de mapeo y autorizadores de Lambda (anteriormente conocidos como autorizadores personalizados). Cuando actualiza la API, puede implementarla de nuevo asociando una nueva etapa a la implementación existente. Explicaremos cómo se crea un etapa en the section called “Configuración de una etapa”.

Temas

- Creación de una implementación
- Próximos pasos para la implementación de la API

Creación de una implementación

En los procedimientos siguientes se muestra cómo crear una implementación para una API de REST.

Consola de administración de AWS

Debe haber creado una API de REST antes de implementarla por primera vez. Para obtener más información, consulte Desarrollo de las API de REST en API Gateway.

La consola de API Gateway le permite implementar una API creando una implementación y asociándola a una etapa nueva o una existente.

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación APIs, elija la API que desea implementar.
3. En el panel Resources (Recursos), elija Deploy API (Implementar API).
4. En Etapa, seleccione una de las siguientes opciones:
   a. Para crear una nueva etapa, seleccione Nueva etapa y, a continuación, ingrese un nombre en Nombre de la etapa. Si lo desea, puede proporcionar una descripción de la implementación en Descripción de la implementación.
   b. Para elegir una etapa existente, seleccione el nombre de la etapa en el menú desplegable. Es posible que desee proporcionar una descripción de la nueva implementación en Descripción de la implementación.
   c. Para crear una implementación que no esté asociada a una etapa, seleccione Sin etapa. Más adelante, puede asociar esta implementación a una etapa.
5. Elija Deploy (Implementar).

AWS CLI

Cuando se crea una implementación, se crea una instancia del recurso Deployment.

El siguiente comando create-deployment permite crear una implementación:

```
aws apigateway create-deployment --rest-api-id rest-api-id
```

No podrá invocar la API hasta haber asociado esta implementación a una etapa. Para hacerlo con una etapa existente, actualice la propiedad deploymentId de la etapa con el ID de la

implementación que acaba de crear. El siguiente comando update-stage actualiza la etapa con una nueva implementación. En la consola, esto se denomina Implementación activa.

```
aws apigateway update-stage \
   --rest-api-id rest-api-id \
   --stage-name 'stage-name' \
   --patch-operations op='replace',path='/deploymentId',value='deployment-id'
```

Cuando crea la implementación, también puede asociarla a una nueva etapa al mismo tiempo. El siguiente comando create-deployment crea una nueva implementación y la asocia a una nueva etapa denominada `beta`:

```
aws apigateway create-deployment \
   --rest-api-id rest-api-id \
   --stage-name beta
```

Para volver a implementar una API, realice los mismos pasos. Puede reutilizar la misma etapa.

Próximos pasos para la implementación de la API

A continuación, se indican los pasos siguientes para la implementación de la API.

Modificación de la configuración de etapa

Una vez que se implementa una API, puede modificar la configuración de las etapas para habilitar o deshabilitar el almacenamiento en caché, el registro o los límites de solicitudes de la API. También puede elegir un certificado de cliente para que el backend autentique a API Gateway y definir variables de etapa para pasar el contexto de implementación a la integración de la API en el tiempo de ejecución. Para obtener más información, consulte Modificación de la configuración de etapa

Después de modificar la configuración de la etapa, es necesario volver a implementar la API para que los cambios surtan efecto.

> **Note**
>
> Si la configuración actualizada, como habilitar el registro, requiere un nuevo rol de IAM, puede agregar el rol de IAM necesario sin tener que volver a implementar la API. No obstante, el nuevo rol de IAM podría tardar unos minutos en empezar a funcionar. Antes

de que esto ocurra, los registros de seguimiento de las llamadas a la API no se registran aunque haya habilitado la opción de registro.

Elección de diferentes combinaciones de etapas de implementación

Como una implementación representa una snapshot de API y una etapa define una ruta en una snapshot, puede elegir diferentes combinaciones de etapas de implementación para controlar la forma en que los usuarios llaman a diferentes versiones de la API. Esto resulta útil, por ejemplo, cuando desea restaurar el estado de la API a una implementación anterior o combinar una "ramificación privada" de la API con una pública.

El siguiente procedimiento muestra cómo hacer esto con el editor de etapas en la consola de API Gateway. Se supone que debe haber implementado una API más de una vez.

1. Si aún no está en el panel Etapas, en el panel de navegación principal, elija Etapas.
2. Seleccione la etapa que desea actualizar.
3. En la pestaña Historial de implementación, seleccione la implementación que desea que utilice la etapa.
4. Elija Cambiar implementación activa.
5. Confirme que desea cambiar la implementación activa y elija Cambiar implementación activa en el cuadro de diálogo Crear implementación activa.

Transfiera los datos específicos de la implementación a su API.

Para una implementación, puede definir o modificar variables de etapa para pasar datos específicos de la implementación a la integración de la API en tiempo de ejecución. Puede hacer esto en la pestaña Stage Variables (Variables de etapa) en Stage Editor (Editor de etapas). Para obtener más información, consulte las instrucciones de Uso de variables de etapa para una API de REST en API Gateway.

Configuración de una etapa para una API de REST en API Gateway

Una etapa es una referencia específica de una implementación, lo que equivale a una instantánea de la API. Puede utilizar una etapa para administrar y optimizar una implementación concreta. Por ejemplo, puede configurar los ajustes de la etapa para habilitar el almacenamiento en caché, personalizar la limitación controlada de solicitudes, configurar los registros, definir variables de etapa

o asociar un lanzamiento canary para realizar pruebas. En la siguiente sección se muestra cómo crear y configurar la etapa.

Creación de una nueva etapa

Después de la implementación inicial, puede añadir más etapas y asociarlas con las implementaciones existentes. Puede utilizar la consola de API Gateway para crear una nueva etapa o puede elegir una etapa existente al implementar una API. En general, puede agregar una nueva etapa a la implementación de una API antes de volver a implementar la API. Para crear una nueva etapa con la consola de API Gateway, siga estos pasos:

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST.
3. En el panel de navegación principal, elija Etapas en una API.
4. En el panel de navegación Etapas, elija Crear etapa.
5. En Nombre de etapa, ingrese un nombre; por ejemplo, **`prod`**.

   > **Note**
   >
   > Los nombres de etapas solo pueden contener caracteres alfanuméricos, guiones y caracteres de subrayado. La longitud máxima es de 128 caracteres.

6. (Opcional). En Descripción, ingrese una descripción de la etapa.
7. En Implementación, seleccione la fecha y la hora de la implementación de la API existente que vaya a asociar con esta etapa.
8. En Configuración adicional, puede especificar ajustes adicionales para su etapa.
9. Elija Crear etapa.

Modificación de la configuración de etapa

Después de una implementación correcta de una API, la etapa se rellena con la configuración predeterminada. Puede utilizar la consola o la API de REST de API Gateway para cambiar las opciones de configuración de las etapas, incluido el almacenamiento en caché de la API y el registro. Si ha modificado el punto de conexión predeterminado de la API de REST, cuando actualice una etapa, la modificación se propagará a todas las etapas de la API. En los siguientes pasos, se indica cómo hacer esto con el Editor de etapas de la consola de API Gateway.

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST.
3. En el panel de navegación principal, elija Etapas en una API.
4. En el panel Etapas, seleccione el nombre de la etapa.
5. En la sección Detalles de la etapa, elija Editar.
6. (Opcional) En Descripción de etapa, escriba una descripción.
7. En Configuración adicional, modifique la siguiente configuración:

Configuración de caché

Para habilitar el almacenamiento en caché de la API para la etapa, active Aprovisionar caché de API. A continuación, configure Almacenamiento en caché de nivel de método predeterminado, Capacidad de caché, Cifrar datos de caché, Tiempo de vida (TTL) de caché y los requisitos de la invalidación de caché para cada clave.

El almacenamiento en caché no está activo hasta que se active el almacenamiento en caché en el nivel de método predeterminado o se active la caché en el nivel de método para un método específico.

Para obtener más información acerca de la configuración de la caché, consulte Configuración de caché para las API de REST en API Gateway.

> **Note**
>
> Si habilita el almacenamiento en caché de la API para una etapa de API, se podrían generar cargos en su cuenta de AWS por el almacenamiento en caché de la API. El almacenamiento en caché no está disponible para la capa gratuita de AWS.

Configuración de limitación

Para definir los objetivos de limitación en el nivel de etapa para todos los métodos asociados a esta API, active Limitación.

ParaRatio, ingrese un ratio objetivo. Esta es la velocidad, en solicitudes por segundo, a la que se añaden los tokens al bucket de tokens. La velocidad a nivel de etapa no debe ser

superior a la velocidad a nivel de cuenta especificada en Cuotas para configurar y ejecutar una API de REST en API Gateway.

Para Ráfaga, ingrese un ratio de ráfaga objetivo. La velocidad de ráfaga es la capacidad del bucket de tokens. Esto permite realizar más solicitudes durante un período de tiempo que el ratio objetivo. Este ratio de ráfaga a nivel de etapa no debe ser superior al ratio de ráfaga a nivel de cuenta especificado en Cuotas para configurar y ejecutar una API de REST en API Gateway.

> **Note**
>
> Los ratios de limitación controlada no son límites estrictos y se aplican en la medida de lo posible. En algunos casos, los clientes pueden sobrepasar los objetivos establecidos. Para controlar los costos o bloquear el acceso a una API, no confíe en la limitación controlada. Considere la posibilidad de utilizar AWS Budgets para monitorear los costos y AWS WAF para administrar las solicitudes de API.

Configuración de firewall y certificados

Para asociar una ACL web de AWS WAF a la etapa, seleccione una ACL web en la lista desplegable ACL web. Si lo desea, elija Bloquear solicitud de API si no se puede evaluar WebACL (cierre por error).

Para seleccionar un certificado de cliente para su etapa, seleccione un certificado en el menú desplegable Certificado de cliente.

8. Elija Continuar.
9. Revise los cambios y elija Guardar cambios.
10. Para habilitar los registros de Amazon CloudWatch para todos los métodos asociados con esta etapa de esta API de API Gateway, en la sección Registros y seguimiento, elija Editar.

> **Note**
>
> Para habilitar los Registros de CloudWatch, también debe especificar el ARN de un rol de IAM que permita a API Gateway escribir información en los Registros de CloudWatch en nombre del usuario. Para ello, elija Configuración en el panel de navegación principal API. A continuación, ingrese el ARN de un rol de IAM en Rol de registro de CloudWatch.

En escenarios de aplicación comunes, el rol de IAM podría asociar la política administrada de AmazonAPIGatewayPushToCloudWatchLogs.
El rol de IAM también debe incluir la siguiente instrucción de relación de confianza:

JSON

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

Para obtener más información sobre CloudWatch, consulte la guía del usuario de Amazon CloudWatch.

11. Elija un nivel de registro en el menú desplegable Registros de CloudWatch. A continuación, se muestran los niveles de registro:

    - Desactivado: el registro no está activado para esta etapa.
    - Solo errores: el registro solo está habilitado para los errores.
    - Registros de errores e información: el registro está habilitado para todos los eventos.

12. Seleccione Rastreo de datos para que API Gateway informe a CloudWatch del registro de rastreo de datos de la etapa. Esto puede ser útil para solucionar problemas de la API, pero puede dar como resultado el registro de datos confidenciales.

> **Note**
>
> Le recomendamos que no utilice Rastreo de datos para las API de producción.

13. Seleccione Métricas detalladas para que API Gateway informe a CloudWatch acerca de las métricas de la API de `API calls`, `Latency`, `Integration latency`, `400 errors` y `500 errors`. Para obtener más información acerca de CloudWatch, consulte Supervisión básica y supervisión detallada en la Guía del usuario de Amazon CloudWatch.

    > **Important**
    >
    > Se aplicarán cargos en su cuenta por obtener acceso a las métricas de CloudWatch en el nivel de método, pero no a las métricas en el nivel de API o de etapa.

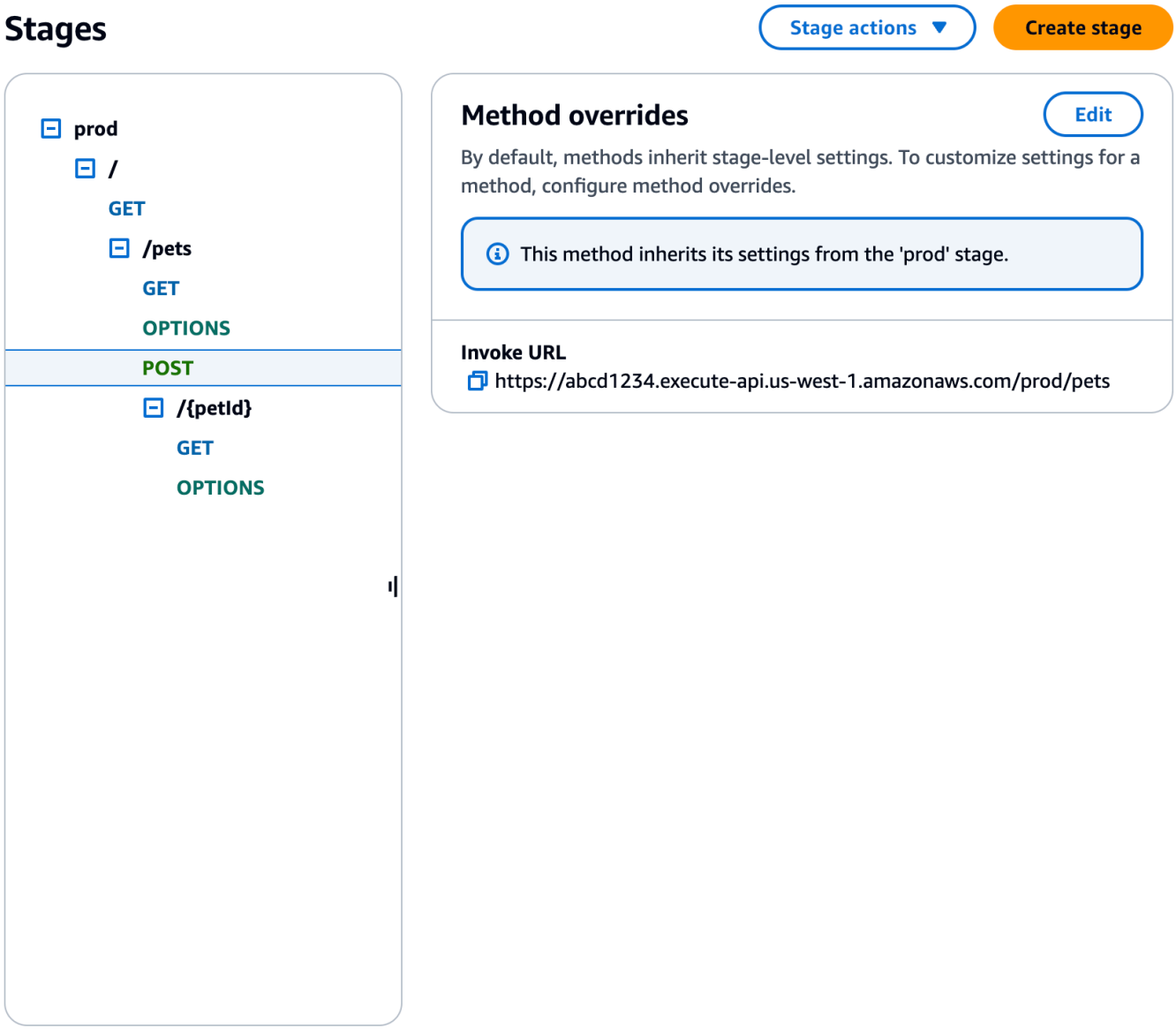
14. Para habilitar el registro de acceso a un destino, active Registro de acceso personalizado.
15. Para ARN del destino de registro de acceso, ingrese el ARN de un grupo de registro o un flujo de Firehose.

    El formato de ARN de Firehose es `arn:aws:firehose:{region}:{account-id}:deliverystream/amazon-apigateway-{your-stream-name}`. El nombre del flujo de Firehose debe ser `amazon-apigateway-{your-stream-name}`.
16. En Formato de registro, ingrese un formato de registro. Para obtener más información sobre ejemplos de formatos de registro, consulte the section called “Formatos de los registro de acceso”.
17. Para habilitar el rastreo de AWS X-Ray para la etapa de la API, seleccione Rastreo de X-Ray. Para obtener más información, consulte Rastreo de las solicitudes de usuario a las API de REST mediante X-Ray en API Gateway.
18. Elija Guardar cambios. Vuelva a implementar su API para que la nueva configuración se aplique.

Anule la configuración a nivel de etapa

Después de personalizar la configuración de nivel de etapa, puede anularla para cada método de API. Algunas de estas opciones pueden dar lugar a cargos adicionales en la Cuenta de AWS.

1. Para configurar las anulaciones de métodos, expanda la etapa bajo el panel de navegación secundario y, a continuación, elija un método.



2. A continuación, en Anulaciones de métodos, elija Editar.
3. Para activar la configuración de CloudWatch a nivel de método, en Registros de CloudWatch, seleccione un nivel de registro.
4. Para activar el registro de rastreo de datos en su método, seleccione Rastreo de datos.

> **Note**
>
> Le recomendamos que no utilice Rastreo de datos para las API de producción.

5. Para activar las métricas detalladas a nivel de método, seleccione Métricas detalladas. Se aplicarán cargos en su cuenta por obtener acceso a las métricas de CloudWatch en el nivel de método, pero no a las métricas en el nivel de API o de etapa.
6. Para activar la limitación en el nivel de método, seleccione Limitación. Introduzca las opciones de nivel de método adecuadas. Para obtener más información sobre limitación controlada, consulte the section called “Limitación controlada”.
7. Para configurar la caché en el nivel de método, seleccione Habilitar caché de métodos. Esta configuración no se verá afectada si en Detalles de la etapa cambia la configuración predeterminada del almacenamiento en caché en el nivel de método.
8. Seleccione Save.

Agregación de una API de REST de API Gateway como destino para la puerta de enlace de Amazon Bedrock AgentCore

Una puerta de enlace de Amazon Bedrock AgentCore proporciona a los desarrolladores de agentes de IA una forma segura de exponer las API de REST de API Gateway como herramientas compatibles con el protocolo de contexto para modelos (MCP). La puerta de enlace AgentCore utiliza los objetivos para definir las herramientas. Cuando agrega la etapa como un destino, la puerta de enlace se convierte en una única URL de MCP que permite el acceso de un agente a las herramientas. Para obtener más información, consulte Etapas de la API de REST de API Gateway como destinos en la Guía para desarrolladores de la puerta de enlace de Amazon Bedrock AgentCore.

Los destinos de API Gateway conectan la puerta de enlace AgentCore con las etapas de las API de REST. Puede incluir la etapa completa como objetivo o seleccionar recursos. Tras crear el destino de API Gateway, la puerta de enlace AgentCore traduce las solicitudes de MCP entrantes en solicitudes HTTP y gestiona el formato de las respuestas. Los clientes de MCP pueden recuperar la documentación de la API mediante el método `tools/list` e invocar las API mediante el método `tools/call`.

Consideraciones

Es posible que las siguientes consideraciones afecten al uso de la agregación de una etapa como destino a una puerta de enlace AgentCore:

- Ya debe tener una puerta de enlace AgentCore.
- Solo se admiten API de REST públicas.

- El punto de conexión predeterminado de la API no se puede desactivar.
- Cada método de la API debe tener un nombre de operación definido para ello o debe crear una invalidación de nombre cuando agregue la etapa como destino. Este nombre se utiliza como el nombre de la herramienta que utilizan los agentes para interactuar con el método.
- Puede usar los tipos de proveedor de credenciales `API_KEY`, `NO_AUTH` o `GATEWAY_IAM_ROLE` para la autenticación saliente a fin de permitir que la puerta de enlace acceda a la API. El proveedor de credenciales `API_KEY` lo define la puerta de enlace AgentCore. Puede usar la clave de API Gateway existente. Para obtener más información, consulte Configuración de la autenticación saliente.
- Si utiliza un autorizador de Lambda o un grupo de usuarios de Amazon Cognito para controlar el acceso a la API, los clientes de MCP no pueden acceder.
- La API debe estar en la misma cuenta y región que la puerta de enlace AgentCore.

Agregación de una etapa de una API como destino para una puerta de enlace AgentCore

En el siguiente procedimiento se muestra cómo agregar una etapa de una API como destino para una puerta de enlace AgentCore.

Agregación de una etapa de una API como destino para una puerta de enlace AgentCore

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST que se implemente en una etapa.
3. En el panel de navegación principal, elija Etapas.
4. Elija Acciones de etapa y, a continuación, elija Crear destino de MCP.
5. Para Puerta de enlace AgentCore, seleccione una puerta de enlace AgentCore.
6. Para Nombre de destino, escriba un nombre de destino.
7. Para Descripción de destino, ingrese una descripción.
8. Conserve la API y la etapa proporcionados.
9. Para seleccionar los recursos de la API, seleccione los recursos de la API a los que pueden acceder los agentes que utilizan la puerta de enlace AgentCore.

   Si no selecciona un recurso, el agente no podrá ver la documentación ni invocar el punto de conexión.
10. La combinación del recurso y el método son las operaciones de la herramienta. Si la operación no tiene un nombre, cree una invalidación de nombre.

También puede definir un nombre de operación para un método al crearlo.

11. Para configuración de autenticación saliente, elija rol de IAM, sin autorización o clave de la API.
12. Elija Crear destino.

Para ver todas las puertas de enlace AgentCore que tienen acceso a las API, elija la sección de destinos de MCP en el panel de navegación principal. En esta sección, puede crear un destino de MCP para cualquier API de la región implementada en una etapa. Elija Crear un destino de MCP y siga los pasos anteriores.

También puede ver las herramientas disponibles para el destino y editarlo en la consola de puerta de enlace AgentCore. Para obtener más información, consulte Agregación de destinos a una puerta de enlace AgentCore existente.

Eliminación de una etapa

Cuando ya no necesite una etapa, puede eliminarla para evitar pagar por recursos que no utiliza. En los pasos siguientes se enseña cómo utilizar la consola de API Gateway para eliminar una etapa.

> **Warning**
>
> La eliminación de una etapa puede provocar que parte o la totalidad de la API correspondiente quede inservible para los intermediarios de la API. La eliminación de una etapa no se puede deshacer, pero puede volver a crear la etapa y asociarla a la misma implementación.

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST.
3. En el panel de navegación principal, elija Etapas.
4. En el panel Etapas, elija la etapa que desea eliminar y, a continuación, elija Acciones de etapa, Eliminar etapa.
5. Cuando se le pregunte, escriba **`confirm`** y, a continuación, elija Eliminar.

Configuración de etiquetas para una etapa de API en API Gateway

API Gateway le permite agregar una etiqueta a una etapa de API, quitarla de la etapa o consultarla. Para ello, puede utilizar la consola de API Gateway, la AWS CLI o un SDK o la API REST de API Gateway.

Una etapa también puede heredar etiquetas de su API REST principal. Para obtener más información, consulte the section called “Herencia de etiquetas en la API V1 de Amazon API Gateway”.

Para obtener más información acerca de las etiquetas para recursos de API Gateway, consulte Etiquetado.

Temas

- Configuración de etiquetas para una etapa de API con la consola de API Gateway
- Configuración de etiquetas para una etapa de API a través de la AWS CLI
- Configuración de etiquetas para una etapa de API con la API de REST de API Gateway

Configuración de etiquetas para una etapa de API con la consola de API Gateway

En el procedimiento siguiente, se describe cómo configurar etiquetas en una etapa de API.

Para configurar etiquetas para una etapa de API con la consola de API Gateway

1. Inicie sesión en la consola de API Gateway.
2. Seleccione una API existente o cree una nueva que contenga recursos, métodos y las integraciones correspondientes.
3. Seleccione una etapa o implemente la API en una nueva etapa.
4. En el panel de navegación principal, elija Etapas.
5. Elija la pestaña Etiquetas. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña.
6. Elija Administrar etiquetas.
7. En Editor de etiquetas, elija Agregar etiqueta. Escriba una clave de etiqueta (por ejemplo, `Department`) en el campo Key (Clave) y escriba un valor para la etiqueta (por ejemplo, `Sales`) en el campo Value (Valor). Elija Guardar para guardar la etiqueta.
8. Si es necesario, repita el paso 5 para agregar más etiquetas a la etapa de API. El número máximo de etiquetas por etapa es 50.

9. Para eliminar una etiqueta de la etapa, haga clic en Eliminar.
10. Si la API se ha implementado anteriormente en la consola de API Gateway, usted tiene que volver a implementarla para que los cambios surtan efecto.

Configuración de etiquetas para una etapa de API a través de la AWS CLI

Puede configurar etiquetas para una etapa de API con AWS CLI mediante el comando create-stage o el comando tag-resource. Puede eliminar una o varias etiquetas de una etapa de API con el comando untag-resource.

El siguiente comando create-stage permite agregar una etiqueta al crear una etapa `test`:

```
aws apigateway create-stage --rest-api-id abc1234 --stage-name test --description
 'Testing stage' --deployment-id efg456 --tag Department=Sales
```

El siguiente tag-resource permite agregar una etiqueta a una etapa `prod`:

```
aws apigateway tag-resource --resource-arn arn:aws:apigateway:us-east-2::/
restapis/abc123/stages/prod --tags Department=Sales
```

El siguiente comando untag-resource permite eliminar la etiqueta `Department=Sales` de la etapa `test`:

```
aws apigateway untag-resource --resource-arn arn:aws:apigateway:us-east-2::/
restapis/abc123/stages/test --tag-keys Department
```

Configuración de etiquetas para una etapa de API con la API de REST de API Gateway

Puede configurar etiquetas para una etapa de API mediante la API de REST de API Gateway con uno de los siguientes procedimientos:

- Llame a `tags:tag` para etiquetar una etapa de API.
- Llame a `tags:untag` para eliminar una o varias etiquetas de una etapa de API.
- Llame a `stage:create` para agregar una o varias etiquetas a una etapa de API que vaya a crear.

También puede llamar a `tags:get` para describir las etiquetas en una etapa de API.

Etiquetado de una etapa de API

Una vez que haya implementado una API (`m5zr3vnks7`) en una etapa (`test`), etiquete la etapa llamando a `tags:tag`. El nombre de recurso de Amazon (ARN) de la etapa correspondiente (`arn:aws:apigateway:us-east-1::/restapis/m5zr3vnks7/stages/test`) debe estar codificado como URL (`arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis%2Fm5zr3vnks7%2Fstages%2Ftest`).

```
PUT /tags/arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis%2Fm5zr3vnks7%2Fstages%2Ftest

{
  "tags" : {
    "Department" : "Sales"
  }
}
```

También puede utilizar la solicitud anterior para actualizar una etiqueta a un nuevo valor.

Puede agregar etiquetas a una etapa cuando llame a `stage:create` para crear la etapa:

```
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"
  }
}
```

Eliminación del etiquetado de una etapa de API

Para eliminar la etiqueta `Department` de la etapa, llame a `tags:untag`:

```
DELETE /tags/arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis%2Fm5zr3vnks7%2Fstages
%2Ftest?tagKeys=Department
Host: apigateway.us-east-1.amazonaws.com
Authorization: ...
```

Para eliminar varias etiquetas, utilice una lista separada por comas de claves de etiqueta en la expresión de consulta; por ejemplo, `?tagKeys=Department,Division,...`.

Descripción de las etiquetas de una etapa de API

Para describir las etiquetas existentes en una determinada etapa, llame a `tags:get`:

```
GET /tags/arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis%2Fm5zr3vnks7%2Fstages
%2Ftags
Host: apigateway.us-east-1.amazonaws.com
Authorization: ...
```

La respuesta correcta será similar a la que se muestra a continuación:

```
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"
    }
}
```

Uso de variables de etapa para una API de REST en API Gateway

Las variables de etapa son pares clave-valor que puede definir como atributos de configuración asociados a una etapa de implementación de una API de REST. Actúan como variables de entorno y se pueden usar en la configuración y las plantillas de asignación de la API. Con las etapas de implementación de API Gateway, puede administrar varias etapas de versión para cada API y usar variables de etapa para configurar una etapa de implementación de API para interactuar con distintos puntos de conexión de backend.

Las variables de etapa no están previstas para su uso con datos confidenciales (por ejemplo, credenciales). Para transferir información confidencial a las integraciones, utilice un autorizador de AWS Lambda. Puede pasar datos confidenciales a integraciones en la salida del autorizador de Lambda. Para obtener más información, consulte the section called “Salida de un autorizador de Lambda de API Gateway”.

Casos de uso de variables de etapa

A continuación, se indican los casos de uso de las variables de etapa.

Especificación de un punto de conexión de backend diferente

- La API puede enviar una solicitud `GET` como un proxy HTTP al host web del backend). Puede usar una variable de etapa de modo que, cuando los intermediarios de la API invoquen su punto de conexión de producción, API Gateway llama a `example.com`. A continuación, cuando los intermediarios de la API invocan la etapa beta, API Gateway llama a un host web diferente, como `beta.example.com`. Asimismo, se pueden utilizar variables de etapa para especificar otro nombre de función de AWS Lambda para cada etapa de la API. No puede usar una variable de etapa para establecer un punto de conexión de integración diferente, como dirigir la solicitud `GET` a una integración de proxy HTTP en una etapa y a una integración de proxy Lambda en otra etapa.

  A la hora de especificar el nombre de una función de Lambda como un valor de variable de etapa, debe configurar manualmente los permisos en la función de Lambda. Al especificar una función de Lambda en la consola de API Gateway, aparecerá un comando de la AWS CLI para configurar los permisos adecuados. También puede usar el siguiente comando de la AWS CLI para hacerlo.

```
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
```

Transferencia de la información mediante plantillas de asignación

> Puede tener acceso a las variables de etapa de las plantillas de asignación o transferir parámetros de configuración al backend de AWS Lambda o HTTP. Por ejemplo, es posible que desee volver a utilizar la misma función de Lambda para varias etapas de la API, pero la función debe leer los datos de una tabla de Amazon DynamoDB diferente según la etapa. En las plantillas de mapeo que generan la solicitud de la función de Lambda, puede utilizar variables de etapa para pasar el nombre de la tabla a Lambda.

Para usar una variable de etapa, primero configure una variable de etapa y, a continuación, asígnele un valor. Por ejemplo, para personalizar el punto de conexión de integración de HTTP, primero cree la variable de etapa `url` y, a continuación, en la solicitud de integración de la API, especifique el valor de la variable de etapa, **`http://${stageVariables.url}`**. Este valor indica a API Gateway que sustituya la variable de etapa `${}` en el tiempo de ejecución según la etapa que esté ejecutando la API. Para obtener más información, consulte the section called “Configuración de variables de etapa para las API de REST en API Gateway”.

Configuración de variables de etapa para las API de REST en API Gateway

En esta sección se muestra cómo diferentes configurar variables de etapa para dos etapas de implementación de una API de muestra mediante la consola de Amazon API Gateway. Para entender cómo usar las variables de etapa en API Gateway, le recomendamos que siga todos los procedimientos de esta sección.

Requisitos previos

Antes de comenzar, asegúrese de que se cumplen los siguientes requisitos previos:

- Debe tener una API disponible en API Gateway. Siga las instrucciones en Desarrollo de las API de REST en API Gateway.
- Debe haber implementado la API al menos una vez. Siga las instrucciones en Implementación de las API de REST en API Gateway.

- Debe haber creado la primera etapa de una API implementada. Siga las instrucciones en Creación de una nueva etapa.

Invocación de un punto de conexión HTTP a través de una API con una variable de etapa

En este procedimiento se describe cómo crear una variable de etapa para un punto de conexión HTTP y dos etapas para la API. Además, creará las variables de etapa `url`, `stageName` y `function` que se utilizan en los siguientes procedimientos de esta sección.

Invocación de un punto de conexión HTTP a través de una API con una variable de etapa

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Cree una API y cree un método `GET` en el recurso raíz de la API. Establezca el tipo de integración en HTTP y la URL del punto de conexión en **`http://${stageVariables.url}`**.
3. Implemente la API en una nueva etapa llamada **`beta`**.
4. En el panel de navegación principal, elija Etapas y, a continuación, elija la etapa beta.
5. En la pestaña Variables de etapa, elija Editar.
6. Elija Agregar variable de etapa.
7. En Nombre, escriba **`url`**. En Valor, ingrese **`httpbin.org/get`**.
8. Elija Agregar variable de etapa y, a continuación, haga lo siguiente:

   En Nombre, escriba **`stageName`**. En Valor, ingrese **`beta`**.
9. Elija Agregar variable de etapa y, a continuación, haga lo siguiente:

   En Nombre, escriba **`function`**. En Valor, ingrese **`HelloWorld`**.
10. Seleccione Save.
11. Ahora cree una segunda etapa. En el panel de navegación Etapas, elija Crear etapa. En Stage name (Nombre de etapa), escriba **`prod`**. Seleccione una implementación reciente en Implementación y, a continuación, elija Crear etapa.
12. Al igual que con la etapa beta, establezca las mismas tres variables de etapa (url, stageName y function) en diferentes valores (**`petstore-demo-endpoint.execute-api.com/petstore/pets`**, **`prod`** y **`HelloEveryone`**), respectivamente.
13. En el panel de navegación Stages (Etapas), elija beta. En Detalles de la etapa, elija el icono de copiar para copiar la URL de invocación de la API y, a continuación, ingrese la URL de

invocación de la API en un navegador web. Comenzará la solicitud `GET` de la etapa beta en el recurso raíz de la API.

> **Note**
>
> El enlace Invoke URL (Invocar URL) apunta al recurso raíz de la API en su etapa beta. Al ingresar la dirección URL en un navegador web se llama al método `GET` de la etapa beta en el recurso raíz. Si los métodos se definen en recursos secundarios y no en el propio recurso raíz, al ingresar la URL en un navegador web se devuelve una respuesta de error `{"message":"Missing Authentication Token"}`. En este caso, debe añadir el nombre de un recurso secundario específico al enlace Invoke URL (Invocar URL).

14. La respuesta obtenida de la solicitud `GET` de la etapa beta se muestra a continuación. También puede verificar el resultado usando un navegador para ir a http://httpbin.org/get. Este valor se asignó a la variable `url` en la etapa beta. Las dos respuestas son idénticas.
15. En el panel de navegación Stages (Etapas), elija la etapa prod. En Detalles de la etapa, elija el icono de copiar para copiar la URL de invocación de la API y, a continuación, ingrese la URL de invocación de la API en un navegador web. Comenzará la solicitud `GET` de la etapa prod en el recurso raíz de la API.
16. La respuesta obtenida de la solicitud `GET` de la etapa prod se muestra a continuación. También puede verificar el resultado usando un navegador para ir a http://petstore-demo-endpoint.execute-api.com/petstore/pets. Este valor se asignó a la variable `url` en la etapa prod. Las dos respuestas son idénticas.

Transferencia de metadatos específicos de la etapa a un backend HTTP

Este procedimiento describe cómo utilizar el valor de una variable de etapa en una expresión de parámetro de consulta para pasar los metadatos específicos en una etapa a un backend HTTP. Utilizaremos la variable de etapa `stageName` declarada en el procedimiento anterior.

Transferencia de metadatos específicos de la etapa a un backend HTTP

1. En el panel de navegación Resource (Recurso), elija el método GET.

   Para agregar un parámetro de cadena de consulta a la URL del método, elija la pestaña Solicitud de método y, a continuación, en la sección Configuración de solicitud de método, elija Editar.

2. Elija Parámetros de cadenas de consulta de URL y haga lo siguiente:

   a. Elija Añadir cadena de consulta.

   b. En Nombre, escriba **`stageName`**.

   c. Mantenga desactivados Obligatorio y Almacenamiento en caché.

3. Seleccione Save.

4. Elija la pestaña Solicitud de integración y, a continuación, en la sección Configuración de solicitud de integración, elija Editar.

5. En URL del punto de conexión, anexe **`?stageName=${stageVariables.stageName}`** al valor de URL definido previamente, de modo que toda la URL del punto de conexión sea **`http://${stageVariables.url}?stageName=${stageVariables.stageName}`**.

6. Elija Implementar API y seleccione la etapa beta.

7. En el panel de navegación principal, elija Etapas. En el panel de navegación Stages (Etapas), elija beta. En Detalles de la etapa, elija el icono de copiar para copiar la URL de invocación de la API y, a continuación, ingrese la URL de invocación de la API en un navegador web.

   > **Note**
   >
   > Utilizamos la etapa beta aquí porque el punto de conexión HTTP, tal como lo especifica la variable `url`, "http://httpbin.org/get", acepta expresiones de parámetro de consulta y las devuelve como el objeto `args` en su respuesta.

8. Obtendrá la siguiente respuesta. Observe que el valor `beta`, asignado a la variable de etapa `stageName`, se transmite al backend como el argumento `stageName`.

```
{
  "args": {
    "stageName": "beta"
  },
  "headers": {
    "Accept": "application/json",
    "Host": "httpbin.org",
    "User-Agent": "AmazonAPIGateway_abcd1234",
    "X-Amzn-Apigateway-Api-Id": "abcd1234",
    "X-Amzn-Trace-Id": "Self=1-abcd-11111111111111;Root=1-11111111-111111111111111"
  },
  "origin": "192.0.2.9",
  "url": "http://httpbin.org/get?stageName=beta"
}
```

Invocación de una función de Lambda a través de una API con una variable de etapa

Este procedimiento describe cómo utilizar una variable de etapa para llamar a una función de Lambda como un backend de la API. Utilice la variable de etapa `function` declarada en Invocación de un punto de conexión HTTP a través de una API con una variable de etapa.

Cuando configure una función de Lambda como el valor de una variable de etapa, utilice el nombre local de la función y, si es posible, incluya su alias o especificación de la versión, como en **`HelloWorld`**, **`HelloWorld:1`** o **`HelloWorld:alpha`**. No utilice el ARN de la función (por ejemplo, **`arn:aws:lambda:us-east-1:123456789012:function:HelloWorld`**). La consola de API Gateway asume que el valor de la variable de etapa de una función de Lambda es el nombre incompleto de la función y expande la variable de etapa especificada en un ARN.

Invocación de una función de Lambda a través de una API con una variable de etapa

1. Cree una función de Lambda llamada **`HelloWorld`** mediante el tiempo de ejecución predeterminado de Node.js. El código debe contener lo siguiente:

   ```
   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...');
   };
   ```

   Para obtener más información sobre cómo crear una función de Lambda, consulte Introducción a la consola de la API de REST.
2. En el panel Recursos, seleccione Crear recurso y, a continuación, haga lo siguiente:

   a. En Ruta de recurso, seleccione /.

   b. En Nombre del recurso, escriba **`lambdav1`**.

   c. Elija Crear recurso.
3. Elija el recurso /lambdav1 y, a continuación, elija Crear método.

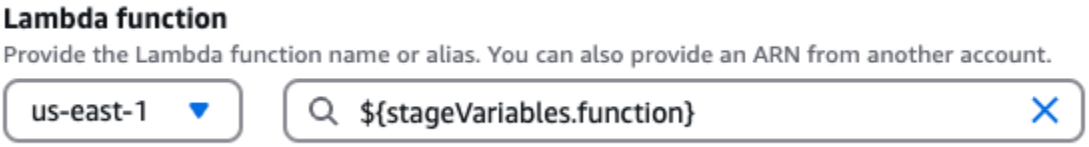
   A continuación, proceda del modo siguiente:

   a. En Tipo de método, seleccione GET.

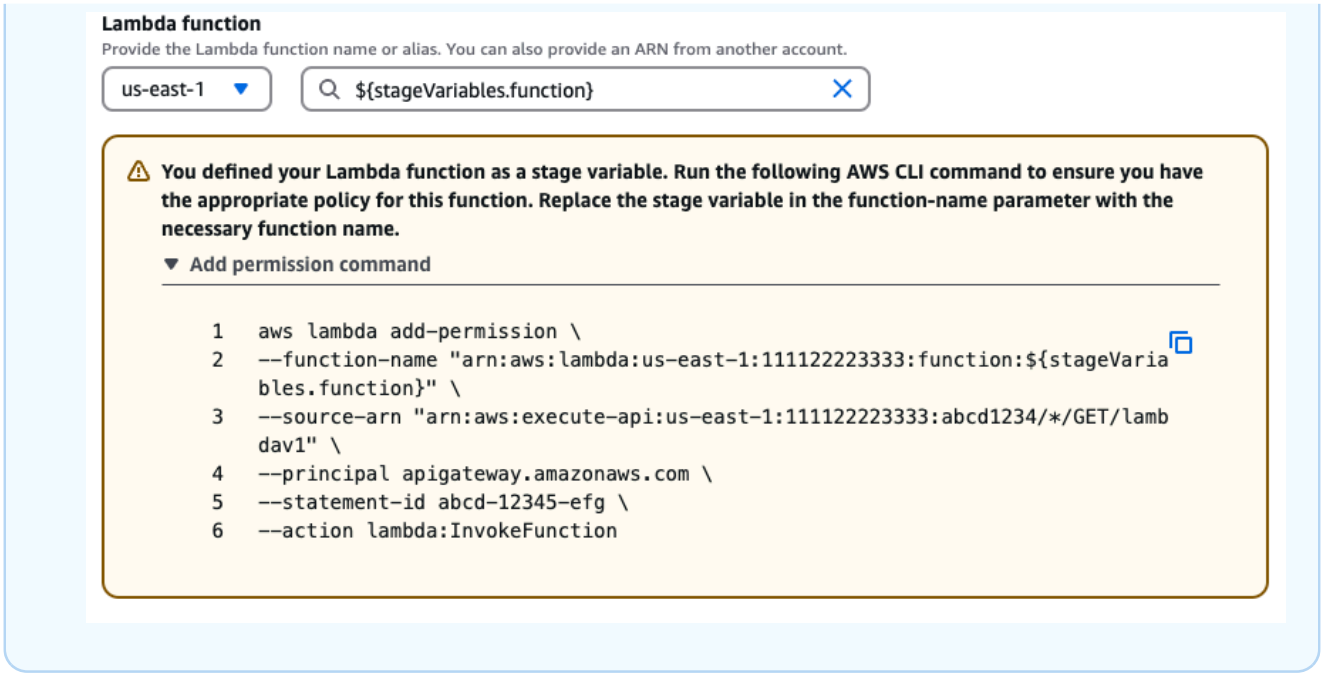
   b. En Tipo de integración, seleccione Función de Lambda.

c. Mantenga desactivada la Integración de proxy Lambda.

d. En Función Lambda, introduzca `${stageVariables.function}`.



> **Tip**
>
> Cuando se le solicite con el comando Agregar permiso, copie el comando add-permission. Ejecute el comando en cada función de Lambda que se vaya a asignar a la variable de etapa `function`. Por ejemplo, si el valor `$stageVariables.function` es `HelloWorld`, ejecute el siguiente comando de la AWS CLI:
>
> ```
> aws lambda add-permission --function-name arn:aws:lambda:us-
> east-1:account-id:function:HelloWorld --source-arn arn:aws:execute-
> api:us-east-1:account-id:api-id/*/GET/lambdav1 --principal
>  apigateway.amazonaws.com --statement-id statement-id-guid --action
>  lambda:InvokeFunction
> ```
>
> En caso contrario, se producirá una respuesta `500 Internal Server Error` al llamar al método. Sustituya `${stageVariables.function}` por el nombre de la función de Lambda asignada a la variable de etapa.



e. Elija Crear método.

4. Implemente la API en las etapas prod y beta.
5. En el panel de navegación principal, elija Etapas. En el panel de navegación Stages (Etapas), elija beta. En Detalles de la etapa, elija el icono de copiar para copiar la URL de invocación de la API y, a continuación, ingrese la URL de invocación de la API en un navegador web. Anexe **`/lambdav1`** a la URL antes de pulsar Intro.

   Obtendrá la siguiente respuesta.

```
"Hello, World! I'm not sure where I'm calling from..."
```

Transferencia de metadatos específicos de una etapa a una función de Lambda a través de una variable de etapa

Este procedimiento describe cómo utilizar una variable de etapa para pasar metadatos de configuración específicos de una etapa a funciones de Lambda. Usted crea un método `POST` y una plantilla de mapeo de entrada para generar la carga mediante la variable de etapa `stageName` declarada anteriormente.

Transferencia de metadatos específicos de la etapa a una función de Lambda a través de una variable de etapa

1. Elija el recurso /lambdav1 y, a continuación, elija Crear método.

A continuación, proceda del modo siguiente:

a. En Tipo de método, seleccione POST.

b. En Tipo de integración, seleccione Función de Lambda.

c. Mantenga desactivada la Integración de proxy Lambda.

d. En Función Lambda, introduzca `${stageVariables.function}`.

e. Cuando se le solicite con el comando Agregar permiso, copie el comando add-permission. Ejecute el comando en cada función de Lambda que se vaya a asignar a la variable de etapa `function`.

f. Elija Crear método.

2. Elija la pestaña Solicitud de integración y, a continuación, en la sección Configuración de solicitud de integración, elija Editar.
3. Elija Plantillas de mapeo y, a continuación, elija Agregar plantilla de mapeo.
4. En Tipo de contenido, ingrese **`application/json`**.
5. En Cuerpo de la plantilla, ingrese la siguiente plantilla:

```
#set($inputRoot = $input.path('$'))
{
    "stageName" : "$stageVariables.stageName"
}
```

> **Note**
>
> En una plantilla de asignación, se debe hacer referencia a una variable de etapa entre comillas (como en `"$stageVariables.stageName"` o `"${stageVariables.stageName}"`). En otros lugares, debe referenciarse sin comillas (como en `${stageVariables.function}`).

6. Seleccione Save.
7. Implemente la API en las etapas beta y prod.
8. Para usar un cliente de API de REST para transferir metadatos específicos de la etapa, haga lo siguiente:

a. En el panel de navegación Stages (Etapas), elija beta. En Detalles de la etapa, elija el icono de copiar para copiar la URL de invocación de la API y, a continuación, ingrese la URL

de invocación de la API en el campo de entrada de un cliente de API de REST. Anexe **/lambdav1** antes de enviar la solicitud.

Obtendrá la siguiente respuesta.

```
"Hello, World! I'm calling from the beta stage."
```

b. En el panel de navegación Etapas, elija prod. En Detalles de la etapa, elija el icono de copiar para copiar la URL de invocación de la API y, a continuación, ingrese la URL de invocación de la API en el campo de entrada de un cliente de API de REST. Anexe **/lambdav1** antes de enviar la solicitud.

Obtendrá la siguiente respuesta.

```
"Hello, World! I'm calling from the prod stage."
```

9. Para usar la característica Pruebas para transferir metadatos específicos de la etapa, haga lo siguiente:

a. En el panel de navegación Recursos, elija la pestaña Pruebas. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña.

b. En function , ingrese **HelloWorld**.

c. En stageName, ingrese **beta**.

d. Seleccione Probar No necesita agregar un cuerpo a su solicitud `POST`.

Obtendrá la siguiente respuesta.

```
"Hello, World! I'm calling from the beta stage."
```

e. Puede repetir los pasos anteriores para probar la etapa Prod. En stageName, ingrese **Prod**.

Obtendrá la siguiente respuesta.

```
"Hello, World! I'm calling from the prod stage."
```

Referencia de variables de etapa de API Gateway para las API de REST en API Gateway

Puede utilizar variables de etapa de API Gateway en los siguientes casos.

Expresiones de mapeo de parámetros

Una variable de etapa se puede usar en una expresión de asignación de parámetros para un parámetro de encabezado de solicitud o respuesta de un método de la API sin ninguna sustitución parcial. En el siguiente ejemplo, se hace referencia a la variable de etapa sin `$` y sin utilizar `{...}`.

- `stageVariables.<variable_name>`

Plantillas de mapeo

Una variable de etapa se puede utilizar en cualquier lugar de una plantilla de asignación, tal y como se muestra en los siguientes ejemplos.

- `{ "name" : "$stageVariables.<variable_name>"}`
- `{ "name" : "${stageVariables.<variable_name>}"}`

URI de integración HTTP

Una variable de etapa se puede utilizar como parte de una URL de integración HTTP, tal y como se muestra en los siguientes ejemplos:

- Una URI completa sin protocol – `http://${stageVariables.<variable_name>}`
- Un dominio complet – `http://${stageVariables.<variable_name>}/resource/operation`
- Un subdomini – `http://${stageVariables.<variable_name>}.example.com/resource/operation`
- Una rut – `http://example.com/${stageVariables.<variable_name>}/bar`
- Una cadena de consult – `http://example.com/foo?q=${stageVariables.<variable_name>}`

AWSURI de integración de

Una variable de etapa se puede utilizar como parte de los componentes de acción o ruta de un URI de AWS, tal y como se muestra en el siguiente ejemplo.

- `arn:aws:apigateway:<region>:<service>:${stageVariables.<variable_name>}`

AWSURI de integración con (funciones de Lambda)

Las variables de etapa pueden utilizarse en lugar de un nombre de función de Lambda o versión/alias, tal y como se muestra en los siguientes ejemplos.

- `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**
>
> Para utilizar una variable de etapa para una función de Lambda, la función debe estar en la misma cuenta que la API. Las variables de etapa no admiten funciones de Lambda entre cuentas.

Grupo de usuarios de Amazon Cognito

Se puede usar una variable de etapa en lugar de un grupo de usuarios de Amazon Cognito como autorizador de `COGNITO_USER_POOLS`.

- `arn:aws:cognito-idp:<region>:<account_id>:userpool/${stageVariables.<variable_name>}`

AWSCredenciales de integración de

Una variable de etapa se puede utilizar como parte del ARN de credenciales de roles/usuarios de AWS, tal y como se muestra en el siguiente ejemplo.

- `arn:aws:iam::<account_id>:${stageVariables.<variable_name>}`

Configuración de una implementación de versión canario de API Gateway

Una versión canario es una estrategia de implementación de software en la que una nueva versión de una API (así como otro software) se implementa con fines de pruebas, mientras que la versión

base sigue implementada como versión de producción para las operaciones normales en la misma etapa. En esta documentación, nos referiremos a la versión base como "versión de producción". No obstante, tiene libertad para aplicar la versión canario de una versión que no sea de producción para realizar pruebas.

Cuando se implementa una versión canario, el tráfico total de la API se separa aleatoriamente entre la versión de producción y la versión canario en un porcentaje preestablecido. Normalmente, el lanzamiento canary recibe un pequeño porcentaje del tráfico de la API, mientras que la versión de producción se ocupa del resto. Las características de la API actualizada solo están visibles al tráfico de la API a través del lanzamiento canary. Puede ajustar el porcentaje de tráfico de canary para optimizar la cobertura o el desempeño de las pruebas.

Si el tráfico del lanzamiento canary es escaso y la selección es aleatoria, muy pocos usuarios se verán afectados en algún momento por los posibles errores de la nueva versión y en ningún caso se verán afectados de forma continuada.

Una vez que las métricas de las pruebas se ajusten a sus requisitos, podrá promover el lanzamiento canary como versión de producción y deshabilitar canary de la implementación. De este modo, las nuevas características pasarán a estar disponibles en la etapa de producción.

Temas

- Implementación de versiones canary en API Gateway
- Creación de una implementación del lanzamiento canary
- Actualización de un lanzamiento canary
- Promoción de un lanzamiento canary
- Desactivación de un lanzamiento Canary

Implementación de versiones canary en API Gateway

En API Gateway, al implementar un lanzamiento canary se utiliza la etapa de implementación de la versión base de una API y se asocia a la etapa un lanzamiento canary de las nuevas versiones (con respecto a la versión base) de la API. La etapa está asociada a la implementación inicial, mientras que el lanzamiento canary está asociado a las implementaciones posteriores. Al principio, tanto la etapa como canary apuntan a la misma versión de la API. En esta sección, utilizaremos indistintamente "etapa" y "versión de producción", así como "canary" y "lanzamiento canary".

Para implementar una API con un lanzamiento canary, cree una implementación del lanzamiento canary agregando las opciones de configuración del lanzamiento canary a la etapa de una

implementación normal. Las opciones de configuración de canary describen el lanzamiento canary subyacente, mientras que la etapa representa la versión de producción de la API en esta implementación. Para agregar la configuración de canary, establezca `canarySettings` en la etapa de implementación y especifique lo siguiente:

- Un ID de implementación (en un principio, será idéntico al ID de la implementación de la versión base definida en la etapa).
- Un porcentaje del tráfico de la API (entre 0,0 y 100,0 inclusives) para el lanzamiento canary.
- Variables de etapa para el lanzamiento canary que puedan sobrescribir las variables de etapa de la versión de producción.
- El uso de la caché de etapas en las solicitudes canary, si se ha definido useStageCache y el almacenamiento en caché de la API está habilitado en la etapa.

Una vez que se habilita un lanzamiento canary, la etapa de implementación no se puede asociar con la implementación que otra versión diferente hasta que el lanzamiento canary se deshabilite y si la configuración de canary se elimina de la etapa.

Cuando se habilitan los registros de ejecución de la API, el lanzamiento canary tiene sus propios registros y métricas, que se generan para todas las solicitudes de canary. Se informa al respecto a un grupo de registro de CloudWatch Logs de etapa de producción, así como a un grupo de registro de CloudWatch Logs específico de canary. Lo mismo sucede con los registros de acceso. Tener registros independientes específicos de canary resulta útil para validar los nuevos cambios de la API y decidir si se van a aceptar las modificaciones y, por tanto, se va a promover el lanzamiento canary como etapa de producción o si, por el contrario, se van a descartar los cambios y se va a revertir el lanzamiento canary como etapa de producción.

El grupo de registros de ejecución de la etapa de producción se llama `API-Gateway-Execution-Logs/{rest-api-id}/{stage-name}`, mientras que el grupo de registros de ejecución del lanzamiento canary se llama `API-Gateway-Execution-Logs/{rest-api-id}/{stage-name}/Canary`. En el caso de los registros de acceso, debe crear un nuevo grupo de registros o seleccionar uno existente. El nombre del grupo de registros de acceso del lanzamiento canary lleva el sufijo `/Canary`.

Un versión canary puede utilizar la caché de etapas (si está habilitada) para guardar las respuestas y utilizar las entradas de esta caché para enviar resultados a las siguientes solicitudes de canary con un tiempo de vida (TTL) preconfigurado.

En la implementación de un lanzamiento canary, la versión de producción y el lanzamiento canary de la API pueden asociarse con la misma versión o con versiones diferentes. Cuando se asocian con diferentes versiones, las respuestas de las solicitudes de producción y de canary se guardan en la caché por separado y la caché de etapas devuelve los resultados correspondientes para los dos tipos de solicitudes. Cuando la versión de producción y el lanzamiento canary están asociados con la misma implementación, la caché de etapas utiliza una única clave de caché para los dos tipos de solicitudes y devuelve la misma respuesta cuando la solicitud es la misma en la versión de producción y el lanzamiento canary.

Creación de una implementación del lanzamiento canary

La implementación del lanzamiento canary se crea cuando la API se implementa con las opciones de configuración canary como datos de entrada, además de con la operación de creación de la implementación.

También puede crear una implementación de lanzamiento canary a partir de una implementación no canary creando una solicitud `stage:update` para agregar la configuración de canary a la etapa.

Cuando crea una implementación de una versión que no es canary, puede especificar un nombre de etapa que no existe. Si no existe la etapa especificada no existe, API Gateway la crea. Sin embargo, no puede especificar una etapa que no existe al crear una implementación canary. Si lo hace, aparecerá un error y API Gateway no creará ninguna implementación del lanzamiento canary.

Puede crear una implementación de lanzamiento canary en API Gateway con la consola de API Gateway, la AWS CLI o un AWS SDK.

Temas

- Creación de una implementación canary con la consola de API Gateway
- Creación de una implementación canary a través de la AWS CLI

Creación de una implementación canary con la consola de API Gateway

Si desea utilizar la consola de API Gateway para crear una implementación de un lanzamiento canary, siga las instrucciones que se indican a continuación:

Para crear la implementación inicial de la versión canario

1. Inicie sesión en la consola de API Gateway.
2. Elija una API de REST existente o cree una nueva API de REST.

3. En el panel de navegación principal, elija Recursos y, a continuación, elija Implementar API. Siga las instrucciones en pantalla que se indican en Implementar API para implementar la API en una nueva etapa.

   Hasta ahora, ha implementado la API en una etapa de versión de producción. A continuación, va a configurar los ajustes de la versión canario en la etapa y, si es necesario, también habilitará el almacenamiento en caché, definirá las variables de etapa o configurará los registros de ejecución o acceso de la API.
4. Para habilitar el almacenamiento en caché de la API o asociar una ACL web de AWS WAF a la etapa, en la sección de Detalles de la etapa, elija Editar. Para obtener más información, consulte the section called “Configuración de caché” o the section called “Asociación de una ACL web de AWS WAF a una etapa de la API de API Gateway mediante la consola de API Gateway”.
5. Para configurar los registros de ejecución o acceso, en la sección Registros y rastreo, elija Editar y siga las instrucciones que aparecen en pantalla. Para obtener más información, consulte Configuración del registro de CloudWatch para las API de REST en API Gateway.
6. Para definir las variables de etapa, elija la pestaña Variables de etapa y siga las instrucciones que aparecen en pantalla para agregar o modificar variables de etapa. Para obtener más información, consulte the section called “Uso de variables de etapa”.
7. Elija la pestaña Canary y, a continuación, elija Crear Canary. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña Canary.
8. En la Configuración de Canary, para Canary, ingrese el porcentaje de solicitudes que se van a desviar al canary.
9. Si lo desea, seleccione Caché de etapa para activar el almacenamiento en caché del lanzamiento canary. La caché no está disponible para el lanzamiento canary hasta que el almacenamiento en caché de la API está habilitado.
10. Para anular las variables de etapa actuales, en Sustitución de Canary, ingrese un nuevo valor de variable de etapa.

Una vez que el lanzamiento canary se inicialice en la etapa de implementación, cambie la API y pruebe los cambios. Puede volver a implementar la API en la misma etapa para que tanto la versión actualizada como la versión base estén accesibles en la misma etapa. En los pasos siguientes, se describe cómo hacerlo.

Para implementar la última versión de la API en un lanzamiento canary

1. Con cada actualización de la API, elija Implementar API.

2. En Implementar API, elija la etapa que contiene un canary en la lista desplegable Etapa de implementación.
3. De forma opcional, ingrese una descripción para Descripción de implementación.
4. Seleccione Deploy (Implementar) para insertar la última versión de la API en el lanzamiento canary.
5. Si lo desea, puede volver a configurar los ajustes, los registros o la configuración de canary de la etapa, tal y como se describe en Para crear la implementación inicial de la versión canario.

Cuando termine, el lanzamiento canary apuntará a la última versión, mientras que la versión de producción seguirá apuntando a la versión inicial de la API. La propiedad canarySettings ahora tiene un nuevo valor deploymentId, mientras que la etapa sigue teniendo el valor deploymentId. En segundo plano, la consola llama a stage:update.

Creación de una implementación canary a través de la AWS CLI

Para crear una implementación canario para una etapa nueva

1. Utilice el siguiente comando create-deployment para crear una implementación con dos variables de etapa, pero sin un canario:

```
aws apigateway create-deployment \
    --variables sv0=val0,sv1=val1 \
    --rest-api-id abcd1234 \
    --stage-name 'prod'
```

   El resultado será similar al siguiente:

```
{
    "id": "du4ot1",
    "createdDate": 1511379050
}
```

2. Utilice el siguiente comando create-deployment para crear una implementación canario en la etapa `prod`:

```
aws apigateway create-deployment \
    --rest-api-id abcd1234 \
```

```
    --canary-settings
 percentTraffic=10.5,stageVariableOverrides={sv1='val2',sv2='val3'},useStageCache=false
 \
    --stage-name 'prod'
```

El resultado será similar al siguiente:

```
{
    "id": "a6rox0",
    "createdDate": 1511379433
}
```

La implementación resultante `id` identifica la versión de prueba de la API del lanzamiento canary. Como resultado, la etapa asociada está habilitada para canary.

3. (Opcional) Utilice el siguiente comando get-stage para ver la representación de la etapa:

```
aws apigateway get-stage --rest-api-id acbd1234 --stage-name prod
```

A continuación, se muestra una representación de `Stage` como la salida del comando:

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

```
}
```

En este ejemplo, la versión de la API utilizará las variables de etapa `{"sv0":val0", "sv1":val1"}`, mientras que la versión de prueba utilizará `{"sv1":val2", "sv2":val3"}`. Tanto la versión de producción como el lanzamiento canary utilizan la variable de etapa `sv1`, pero con diferentes valores: `val1` y `val2`, respectivamente. La variable de etapa `sv0` solo se utiliza en la versión de producción, mientras que `sv2` solo se utiliza en el lanzamiento canary.

También puede crear una implementación de versión canario a partir de una implementación normal existente actualizando la etapa para habilitar un canario.

Para crear una implementación de versión canario a partir de una implementación existente

1. Utilice el comando create-deployment para crear una implementación sin un canario:

```
aws apigateway create-deployment \
    --variables sv0=val0,sv1=val1 \
    --rest-api-id abcd1234 \
    --stage-name 'beta'
```

2. Utilice el comando update-stage para actualizar la etapa y habilitar un canario:

```
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"
        }]'
```

El resultado será similar al siguiente:

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

Como hemos habilitado un canario en una versión existente de la API, tanto la versión de producción (`Stage`) como la versión canario (`canarySettings`) apuntan a la misma implementación. Después de cambiar la API y e implementarla de nuevo en esta etapa, la nueva versión estará en el lanzamiento canary, mientras que la versión base seguirá en la versión de producción. Esto se pone de manifiesto a medida que evoluciona la etapa, cuando el valor de `deploymentId` del lanzamiento canary se actualiza al nuevo `id` de la implementación y el valor de `deploymentId` de la versión de producción se mantiene sin cambios.

Actualización de un lanzamiento canary

Una vez que se implemente el lanzamiento canary, es posible que quiera ajustar el porcentaje de tráfico de canary o que desee habilitar o deshabilitar el uso de la caché de etapas para optimizar el rendimiento de las pruebas. También puede modificar las variables de etapa que se utilizan en el lanzamiento canary cuando se actualiza el contexto de ejecución. Para realizar este tipo de actualizaciones, llame a la operación stage:update utilizando valores nuevos en canarySettings.

Puede actualizar un lanzamiento canary con la consola de API Gateway, el comando update-stage de la AWS CLI o un AWS SDK.

Temas

- Actualización de un lanzamiento canary con la consola de API Gateway
- Actualización de un lanzamiento canary a través de la AWS CLI

Actualización de un lanzamiento canary con la consola de API Gateway

Si desea utilizar la consola de API Gateway para actualizar la configuración de canary de una etapa, haga lo siguiente:

Para actualizar la configuración de canary actual

1. Inicie sesión en la consola de API Gateway y elija la API de REST actual.
2. En el panel de navegación principal, elija Etapas y, a continuación, elija la etapa actuala.
3. Elija la pestaña Canary y, a continuación, elija Editar. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña Canary.
4. Actualice Distribución de solicitudes aumentando o reduciendo el porcentaje en un valor comprendido entre 0,0 y 100,0 (ambos incluidos).
5. Active o desactive la casilla de verificación Caché de etapa.
6. Agregue, elimine o modifique las Variables de etapa de Canary.
7. Seleccione Save.

Actualización de un lanzamiento canary a través de la AWS CLI

Para usar la AWS CLI para actualizar un canario, utilice el comando `update-stage` y modifique la operación de parche para cada parámetro del canario.

El siguiente comando update-stage se actualiza si el canario usa la caché de etapas:

```
aws apigateway update-stage \
    --rest-api-id {rest-api-id} \
    --stage-name '{stage-name}' \
    --patch-operations op=replace,path=/canarySettings/useStageCache,value=true
```

El siguiente comando update-stage permite actualizar el porcentaje de tráfico del canario:

```
aws apigateway update-stage \
    --rest-api-id {rest-api-id} \
    --stage-name '{stage-name}' \
    --patch-operations op=replace,path=/canarySettings/percentTraffic,value=25.0
```

El siguiente comando update-stage actualiza las variables de etapa. En el ejemplo se muestra cómo crear una nueva variable de etapas llamada `newVar`, invalidar la variable de etapas `var2` y eliminar la variable de etapas `var1`:

```
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"
      }]'
```

Puede realizar todas las actualizaciones anteriores combinando las operaciones en un solo valor `patch-operations`:

```
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"
      }]'
```

Promoción de un lanzamiento canary

Cuando promocione un lanzamiento de canario, dicha versión reemplaza la configuración de etapa actual. Cuando se promueve un lanzamiento canary, no se deshabilita canary en la etapa. Para deshabilitar canary, debe eliminar la configuración de canary de la etapa. Para promocionar un canario, haga lo siguiente.

- Restablezca el ID de implementación de la etapa con el ID de implementación del lanzamiento canary. De este modo, la instantánea de API de la etapa se actualiza con la instantánea de canary, lo que convierte la versión de prueba también en versión de producción.
- Actualice las variables de etapa con las variables de canary, si hay alguna. De este modo, el contexto de ejecución de la etapa de la API se actualiza con el de canary. Sin esta actualización, la nueva versión de la API puede producir resultados inesperados si la versión de prueba utiliza otras variables de etapa o los valores de las variables de etapa existentes son diferentes.
- Establezca el porcentaje del tráfico de canary en 0,0 %.

Temas

- Promoción de un lanzamiento canary con la consola de API Gateway
- Promoción de un lanzamiento canary a través de la AWS CLI

Promoción de un lanzamiento canary con la consola de API Gateway

Si desea utilizar la consola de API Gateway para promover la implementación de un lanzamiento canary, haga lo siguiente:

Para promocionar una implementación del lanzamiento canary

1. Inicie sesión en la consola de API Gateway y seleccione una API existente en el panel de navegación principal.
2. En el panel de navegación principal, elija Etapas y, a continuación, elija la etapa actuala.
3. Elija la pestaña Canary.
4. Elija Promocionar Canary.
5. Confirme los cambios que desea realizar y elija Promocionar Canary.

Tras realizar la promoción, la versión de producción hace referencia a la misma versión de la API (deploymentId) que el lanzamiento canary. Puede comprobarlo a través de la AWS CLI. Por ejemplo, consulte the section called “Promoción de un lanzamiento canary a través de la AWS CLI”.

Promoción de un lanzamiento canary a través de la AWS CLI

Para promover un lanzamiento canary como versión de producción utilizando los comandos de la AWS CLI, llame al comando `update-stage` para copiar el `deploymentId` asociado a canary en el `deploymentId` asociado a la etapa, para restablecer el porcentaje del tráfico de canary en cero (`0.0`) y para copiar las variables de etapa vinculadas a canary en las variables correspondientes vinculadas a la etapa.

Supongamos que contamos con una implementación de un lanzamiento canary, descrita por una etapa similar a la siguiente:

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

Utilice el siguiente comando update-stage para promocionar el canario:

```
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"
      }]'
```

El resultado será similar al siguiente:

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

La promoción de un lanzamiento de canario según la etapa no desactiva el canario y la implementación sigue siendo una implementación de un lanzamiento de canario. Para convertirla en una implementación de una versión de producción ordinaria, debe desactivar la configuración de canary. Para obtener más información acerca de cómo desactivar la implementación de un lanzamiento canary, consulte the section called “Desactivación de un lanzamiento Canary”.

Desactivación de un lanzamiento Canary

Para desactivar la implementación de un lanzamiento canary, establezca `canarySettings` en null; de esta forma, se eliminará de la etapa.

Puede desactivar la implementación de un lanzamiento canary con la consola de API Gateway, la AWS CLI o un AWS SDK.

Temas

- Desactivación de un lanzamiento canary con la consola de API Gateway
- Desactivación de un lanzamiento canary a través de la AWS CLI

Desactivación de un lanzamiento canary con la consola de API Gateway

Si desea utilizar la consola de API Gateway para desactivar una implementación de lanzamiento canary, siga estos pasos:

Para desactivar la implementación de un lanzamiento Canary

1. Inicie sesión en la consola de API Gateway y elija una API existente en el panel de navegación principal.
2. En el panel de navegación principal, elija Etapas y, a continuación, elija la etapa actuala.
3. Elija la pestaña Canary.
4. Elija Eliminar.
5. Confirme que desea eliminar el lanzamiento canary seleccionando Delete (Eliminar).

Como resultado, la propiedad `canarySettings` pasa a ser `null` y se elimina de la etapa de implementación. Puede comprobarlo a través de la AWS CLI. Por ejemplo, consulte the section called “Desactivación de un lanzamiento canary a través de la AWS CLI”.

Desactivación de un lanzamiento canary a través de la AWS CLI

El siguiente comando update-stage permite desactivar la implementación de la versión canario:

```
aws apigateway update-stage \
    --rest-api-id abcd1234 \
    --stage-name canary \
    --patch-operations '[{"op":"remove", "path":"/canarySettings"}]'
```

El resultado es similar al siguiente:

```
{
    "stageName": "prod",
    "accessLogSettings": {
        ...
    },
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "deploymentId": "nfcn0x",
    "lastUpdatedDate": 1511309280,
    "createdDate": 1511152939,
    "methodSettings": {
```

```
        ...
    }
}
```

Tal como se muestra en el resultado, la propiedad canarySettings ya no está presente en la etapa de una implementación con la versión canario desactivada.

Actualizaciones de las API de REST que requieren reimplementación

Mantener una API equivale a visualizar, actualizar y eliminar las configuraciones existentes de la API. Puede mantener una API con la consola de API Gateway, la AWS CLI, el CloudFormation, un SDK o la API de REST de API Gateway. La actualización de una API implica modificar determinadas propiedades de recursos u opciones de configuración de la API. Las actualizaciones de recursos requieren volver a implementar la API, pero no las actualizaciones de configuración.

En la siguiente tabla, se describen los recursos de la API que requieren volver a implementar la API al actualizarlos.

Recurso	Notas
ApiKey	Para saber cuáles son las propiedades aplicables y las operaciones que se pueden realizar, consulte apikey:update. La actualización requiere volver a implementar la API.
Authorizer	Para saber cuáles son las propiedades aplicables y las operaciones que se pueden realizar, consulte authorizer:update. La actualización requiere volver a implementar la API.
disableExecuteApiEndpoint	La actualización requiere modificar cualquier etapa de la API, como volver a implementar la API en una etapa.
DocumentationPart	Para saber cuáles son las propiedades aplicables y las operaciones que se pueden realizar, consulte documentationpart:update. La actualización requiere volver a implementar la API.
DocumentationVersion	Para saber cuáles son las propiedades aplicables y las operaciones que se pueden realizar, consulte documentationversion:update. La actualización requiere volver a implementar la API.

Recurso	Notas
GatewayResponse	Para saber cuáles son las propiedades aplicables y las operaciones que se pueden realizar, consulte gatewayresponse:update. La actualización requiere volver a implementar la API.
Integration	Para saber cuáles son las propiedades aplicables y las operaciones que se pueden realizar, consulte integration:update. La actualización requiere volver a implementar la API.
IntegrationResponse	Para saber cuáles son las propiedades aplicables y las operaciones que se pueden realizar, consulte integrationresponse:update. La actualización requiere volver a implementar la API.
Método	Para saber cuáles son las propiedades aplicables y las operaciones que se pueden realizar, consulte method:update. La actualización requiere volver a implementar la API.
MethodResponse	Para saber cuáles son las propiedades aplicables y las operaciones que se pueden realizar, consulte methodresponse:update. La actualización requiere volver a implementar la API.
Modelo	Para saber cuáles son las propiedades aplicables y las operaciones que se pueden realizar, consulte model:update. La actualización requiere volver a implementar la API.
RequestValidator	Para saber cuáles son las propiedades aplicables y las operaciones que se pueden realizar, consulte requestvalidator:update. La actualización requiere volver a implementar la API.
Recurso	Para saber cuáles son las propiedades aplicables y las operaciones que se pueden realizar, consulte resource:update. La actualización requiere volver a implementar la API.
RestApi	Para saber cuáles son las propiedades aplicables y las operaciones que se pueden realizar, consulte restapi:update. La actualización requiere volver a implementar la API. Esto incluye la modificación de las políticas de recursos.

Recurso	Notas
VpcLink	Para saber cuáles son las propiedades aplicables y las operaciones que se pueden realizar, consulte vpclink:update. La actualización requiere volver a implementar la API.

En la siguiente tabla, se describen las configuraciones de la API que no requieren volver a implementar la API al actualizarlos.

Configuración	Notas
Cuenta	Para saber cuáles son las propiedades aplicables y las operaciones que se pueden realizar, consulte account:update. La actualización no requiere volver a implementar la API.
Implementación	Para saber cuáles son las propiedades aplicables y las operaciones que se pueden realizar, consulte deployment:update.
DomainName	Para saber cuáles son las propiedades aplicables y las operaciones que se pueden realizar, consulte domainname:update. La actualización no requiere volver a implementar la API.
BasePathMapping	Para saber cuáles son las propiedades aplicables y las operaciones que se pueden realizar, consulte basepathmapping:update. La actualización no requiere volver a implementar la API.
Tipo de dirección IP	La actualización no requiere volver a implementar la API.
Etapa	Para saber cuáles son las propiedades aplicables y las operaciones que se pueden realizar, consulte stage:update. La actualización no requiere volver a implementar la API.
Uso	Para saber cuáles son las propiedades aplicables y las operaciones que se pueden realizar, consulte usage:update. La actualización no requiere volver a implementar la API.

Configuración	Notas
UsagePlan	Para saber cuáles son las propiedades aplicables y las operaciones que se pueden realizar, consulte usageplan:update. La actualización no requiere volver a implementar la API.

Nombre de dominio personalizado para las API de REST públicas en API Gateway

Losnombres de dominio personalizados son direcciones URL más sencillas e intuitivas que puede proporcionar a los usuarios de la API.

Después de implementar la API, usted (y sus clientes) puede invocar la API utilizando la URL base predeterminada con el siguiente formato:

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

donde *api-id* lo genera API Gateway, *región* es la región de AWS y *etapa* lo especifica usted al implementar la API.

La parte del nombre de host de la URL, *api-id*`.execute-api.`*region*`.amazonaws.com`, hace referencia a un punto de conexión de la API. El punto de conexión de la API predeterminado se genera aleatoriamente y no es muy descriptivo.

Con los nombres de dominio personalizados, puede configurar el nombre de host de la API y elegir una ruta base (por ejemplo, `myservice`) para asignarle una URL alternativa. Por ejemplo, una URL base de la API más simple puede ser:

```
https://api.example.com/myservice
```

> **Note**
>
> Para obtener más información sobre los nombres de dominio personalizados para API privadas, consulte the section called “Nombres de dominio personalizados para API privadas”.

Consideraciones

Es posible que las siguientes consideraciones afecten al uso de un nombre de dominio personalizado:

- Puede deshabilitar el punto de conexión predeterminado para la API. Los clientes pueden seguir conectándose al punto de conexión predeterminado, pero recibirán un código de estado `403 Forbidden`.
- Un nombre de dominio personalizado regional se puede asociar con las API de REST y las API de HTTP. Puede utilizar las API de API Gateway versión 2 para crear y administrar nombres de dominio personalizados regionales para las API de REST.
- Un nombre de dominio personalizado debe ser único en una región en todas las cuentas de AWS.
- Puede migrar su nombre de dominio personalizado entre puntos de conexión optimizados para bordes y regionales, pero no puede migrar un dominio personalizado público a un nombre de dominio personalizado privado.
- Debe crear o actualizar el registro de recursos del proveedor de DNS para asignarlo al punto de conexión de la API. Si no se realiza este mapeo, las solicitudes de API vinculadas al nombre de dominio personalizado no pueden llegar a API Gateway.
- Puede admitir un número casi infinito de nombres de dominio sin exceder el límite predeterminado mediante el uso de un certificado comodín. Para obtener más información, consulte the section called “Nombres de dominio personalizados comodín”.
- Puede elegir una política de seguridad para el nombre de dominio personalizado. Para obtener más información, consulte the section called “Elección de una política de seguridad”.
- Para configurar asignaciones de la API con varios niveles, debe usar un nombre de dominio personalizado regional y usar la política de seguridad de TLS 1.2.

Requisitos previos de los nombres de dominio personalizados

A continuación se enumeran los requisitos previos para crear un nombre de dominio personalizado público o privado. Para obtener más información sobre los nombres de dominio personalizados para API privadas, consulte the section called “Nombres de dominio personalizados para API privadas”.

Registrar un nombre de dominio

Debe disponer de un nombre de dominio de Internet registrado para poder crear nombres de dominio personalizados para sus API. Puede registrar su nombre de dominio de Internet con Amazon Route

53 o un registrador de dominios de terceros de su elección. Su nombre de dominio personalizado puede ser el nombre de un subdominio o el dominio raíz (lo que recibe el nombre de “vértice de zona”) de un dominio de Internet registrado.

El nombre de dominio debe seguir la especificación RFC 1035 y puede tener un máximo de 63 octetos por etiqueta y 255 octetos en total.

Especificación del certificado para el nombre de dominio personalizado

Antes de configurar un nombre de dominio personalizado para una API, debe disponer de un certificado SSL/TLS listo en ACM. Si ACM no está disponible en la región de AWS donde va a crear su nombre de dominio personalizado, debe importar un certificado en API Gateway en esa región.

Para importar un certificado SSL/TLS, debe proporcionar el cuerpo del certificado SSL/TLS con formato PEM, su clave privada y la cadena de certificados para el nombre de dominio personalizado.

Los certificados almacenados en ACM se identifican mediante su ARN. Con los certificados emitidos por ACM no tiene que preocuparse de si se exponen datos del certificado confidenciales, como la clave privada. Para utilizar un certificado administrado por AWS para un nombre de dominio, solo tiene que hacer referencia a su ARN.

Si la aplicación utiliza asignación de certificados, también conocido como asignación SSL, para asignar un certificado de ACM, es posible que la aplicación no se pueda conectar al dominio una vez que AWS renueva el certificado. Para obtener más información, consulte Problemas de asignación de certificados en la Guía del usuario de AWS Gestor de certificados.

Nombres de dominio personalizados comodín

Con los nombres de dominio personalizados comodín, puede admitir un número casi infinito de nombres de dominio sin exceder el límite predeterminado. Por ejemplo, puede dar a cada uno de los clientes su propio nombre de dominio, *`customername`*`.example.com`.

Para crear un nombre de dominio personalizado comodín, especifique un comodín (`*`) como primer subdominio de un dominio personalizado que represente todos los subdominios posibles de un dominio raíz.

Por ejemplo, el nombre de dominio personalizado comodín `*.example.com` produce subdominios como `a.example.com`, `b.example.com` y `c.example.com`. Cuando crea el nombre de dominio personalizado comodín, todos los subdominios se enrutan por el modo de enrutamiento del nombre

de dominio comodín. Para enrutar subdominios a diferentes API, puede hacer una de las siguientes acciones:

- Utilizar reglas de enrutamiento para enrutar las solicitudes entrantes a `*.example.com` a diferentes API de REST de destino mediante el encabezado `Host`. Para obtener más información, consulte the section called “Ejemplo 4: reglas de enrutamiento para nombres de dominio comodín”.
- Crear un nombre de dominio para cualquier subdominio que desee enrutar a un punto de conexión diferente. En una misma cuenta de AWS, puede tener `*.example.com` y `a.example.com`.

Puede utilizar las variables de contexto `$context.domainName` y `$context.domainPrefix` para determinar el nombre de dominio que un cliente utilizó para llamar a la API. Para obtener más información acerca de las variables de contexto, consulte Variables para transformaciones de datos para API Gateway.

Para crear un nombre de dominio personalizado comodín, debe proporcionar un certificado emitido por ACM que se haya validado utilizando el método DNS o de validación de correo electrónico.

> **Note**
>
> No se puede crear un nombre de dominio personalizado comodín si una cuenta de AWS diferente ha creado un nombre de dominio personalizado que entra en conflicto con el nombre de dominio personalizado comodín. Por ejemplo, si la cuenta A ha creado `a.example.com`, la cuenta B no puede crear el nombre de dominio personalizado comodín `*.example.com`.
> Si la cuenta A y la cuenta B comparten un propietario, puede contactar con el Centro de soporte de AWS para solicitar una excepción.

Siguientes pasos para nombres de dominio personalizados

A continuación, se muestran los siguientes pasos para los nombres de dominio personalizados.

Siguientes pasos

- Para obtener información sobre cómo configurar su certificado SSL/TLS, consulte the section called “Preparación de certificados en AWS Gestor de certificados”.
- Para obtener información sobre cómo crear un dominio personalizado regional, consulte the section called “Configuración de un nombre de dominio personalizado regional”.

- Para obtener información sobre cómo crear un nombre de dominio personalizado optimizado para la periferia, consulte the section called “Configuración de un nombre de dominio personalizado optimizado para la periferia para en API Gateway”.
- Para obtener información sobre cómo migrar entre nombres de dominio personalizados regionales y optimizados para la periferia, consulte the section called “Migración de nombres de dominio personalizados”.
- Para obtener información sobre cómo conectar etapas de la API a un nombre de dominio personalizado, consulte the section called “Envío de tráfico a las API a través del nombre de dominio personalizado en API Gateway”.
- Para obtener información sobre cómo elegir una política de seguridad para el nombre de dominio personalizado, consulte the section called “Elección de una política de seguridad”.
- Para obtener información sobre cómo desactivar el punto de conexión predeterminado para su nombre de dominio personalizado, consulte the section called “Deshabilitación del punto de conexión predeterminado”.
- Para obtener información sobre cómo utilizar las comprobaciones de estado de Route 53 para controlar la conmutación por error de DNS desde una API de API Gateway, consulte the section called “recuperación ante errores a nivel de DNS”.

Si es la primera vez que crea un nombre de dominio personalizado, le recomendamos que comience con the section called “Preparación de certificados en AWS Gestor de certificados”, para especificar el certificado, y, después, the section called “Configuración de un nombre de dominio personalizado regional”, para crear un nombre de dominio personalizado regional.

Preparación de certificados en AWS Gestor de certificados

Antes de configurar un nombre de dominio personalizado para una API, debe disponer de un certificado SSL/TLS listo en AWS Gestor de certificados. Para obtener más información, consulte la Guía del usuario de AWS Gestor de certificados.

Consideraciones

Las siguientes son consideraciones para su certificado SSL/TLS.

- Si crea un nombre de dominio personalizado optimizado para la periferia, API Gateway usa CloudFront para admitir certificados para nombres de dominio personalizados. Por tanto, los requisitos y las restricciones del certificado SSL/TLS de un nombre de dominio personalizado los dicta CloudFront. Por ejemplo, el tamaño máximo de la clave pública es 2048 y la clave privada

puede ser 1024, 2048 y 4096. El tamaño de la clave pública se determina en función de la entidad de certificación que utilice. Pida a su entidad de certificación que devuelva claves de un tamaño diferente de la longitud predeterminada. Para obtener más información, consulte Proteger el acceso a los objetos y Creación de URL y cookies firmadas.

- Si crea un nombre de dominio personalizado regional, el tamaño máximo de la clave pública es 2048.
- Para usar un certificado de ACM con un nombre de dominio personalizado regional, debe solicitar o importar el certificado en la misma región que la API. El certificado debe cubrir el nombre de dominio personalizado.
- Si desea utilizar un certificado de ACM con un nombre de dominio personalizado optimizado para la periferia, debe solicitar o importar el certificado en la región Este de EE. UU. (Norte de Virginia) - `us-east-1`.
- Debe tener un nombre de dominio registrado, como *`example.com`*. Puede utilizar Amazon Route 53 o un registrador de dominios de terceros acreditado. Para obtener una lista de registradores, consulte Accredited Registrar Directory en el sitio web de ICANN.

Creación o importación de un certificado SSL/TLS en ACM

En los siguientes procedimientos se muestra cómo crear o importar un certificado SSL/TLS para un nombre de dominio.

To request a certificate provided by ACM for a domain name

1. Inicie sesión en la consola de AWS Gestor de certificados.
2. Elija Request a certificate (Solicitar un certificado).
3. En Tipo de certificado, elija Solicitar un certificado público.
4. Elija Siguiente.
5. En Nombre de dominio completo, introduzca un nombre de dominio para la API, por ejemplo, `api.example.com`.
6. También puede seleccionar Add another name to this certificate (Añadir otro nombre a este certificado).
7. En Método de validación, elija un método para validar la propiedad del dominio.
8. En Algoritmo clave, elija un algoritmo de cifrado.
9. Seleccione Solicitar.

10. Para que una solicitud sea válida, un propietario registrado del dominio de Internet debe autorizar la solicitud para que ACM emita el certificado. Si utiliza Route 53 para administrar sus registros de DNS públicos, puede actualizar los registros directamente a través de la consola de ACM.

To import into ACM a certificate for a domain name

1. Obtenga un certificado SSL/TLS codificado en PEM para su nombre de dominio personalizado de una entidad de certificación. Para obtener una lista parcial de entidades de certificación, consulte Mozilla Included CA List.

   a. Genere una clave privada para el certificado y guarde el resultado en un archivo, utilizando el kit de herramientas de OpenSSL en el sitio web de OpenSSL:

   ```
   openssl genrsa -out private-key-file 2048
   ```

   b. Genere una solicitud de firma de certificado (CSR) con la clave privada generada anteriormente, utilizando OpenSSL:

   ```
   openssl req -new -sha256 -key private-key-file -out CSR-file
   ```

   c. Envíe la CSR a la entidad de certificación y guarde el certificado resultante.

   d. Descargue la cadena de certificados de la entidad de certificación.

   > **Note**
   >
   > Si obtiene la clave privada de otra forma y la clave está cifrada, puede utilizar el siguiente comando para descifrar la clave antes de enviarla a API Gateway para que configure un nombre de dominio personalizado.
   >
   > ```
   > openssl pkcs8 -topk8 -inform pem -in MyEncryptedKey.pem -outform pem -
   > nocrypt -out MyDecryptedKey.pem
   > ```

2. Cargue el certificado en AWS Gestor de certificados:

   a. Inicie sesión en la consola de AWS Gestor de certificados.

   b. Seleccione Import a certificate.

c. En Cuerpo del certificado, introduzca o pegue el cuerpo del certificado del servidor con formato PEM de su entidad de certificación. A continuación se muestra un ejemplo abreviado de dicho certificado.

```
-----BEGIN CERTIFICATE-----
EXAMPLECA+KgAwIBAgIQJ1XxJ8Pl++gOfQtj0IBoqDANBgkqhkiG9w0BAQUFADBB
...
az8Cg1aicxLBQ7EaWIhhgEXAMPLE
-----END CERTIFICATE-----
```

d. En Clave privada del certificado, introduzca o pegue la clave privada del certificado con formato PEM. A continuación se muestra un ejemplo abreviado de dicha clave.

```
-----BEGIN RSA PRIVATE KEY-----
EXAMPLEBAAKCAQEA2Qb3LDHD7StY7Wj6U2/opV6Xu37qUCCkeDWhwpZMYJ9/nETO
...
1qGvJ3u04vdnzaYN5WoyN5LFckrlA71+CszD1CGSqbVDWEXAMPLE
-----END RSA PRIVATE KEY-----
```

e. En Cadena de certificados, introduzca o pegue los certificados intermedios con formato PEM y, opcionalmente, el certificado raíz, uno detrás del otro sin líneas en blanco. Si incluye el certificado raíz, la cadena de certificados debe empezar con los certificados intermedios y terminar con el certificado raíz. Utilice los certificados intermedios proporcionados por la entidad de certificación. No incluya certificados intermedios que no estén en la cadena de la ruta de confianza. A continuación se muestra un ejemplo abreviado.

```
-----BEGIN CERTIFICATE-----
EXAMPLECA4ugAwIBAgIQWrYdrB5NogYUx1U9Pamy3DANBgkqhkiG9w0BAQUFADCB
...
8/ifBlIK3se2e4/hEfcEejX/arxbx1BJCHBvlEPNnsdw8EXAMPLE
-----END CERTIFICATE-----
```

Este es otro ejemplo.

```
-----BEGIN CERTIFICATE-----
Intermediate certificate 2
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
Intermediate certificate 1
```

```
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
Optional: Root certificate
-----END CERTIFICATE-----
```

f. Seleccione Siguiente y de nuevo Siguiente.

Una vez que el certificado se haya creado o importado correctamente, anote el ARN del certificado. Lo necesitará cuando configure el nombre de dominio personalizado.

Configuración de un nombre de dominio personalizado regional en API Gateway

Utilice un nombre de dominio personalizado regional para crear una URL base de la API más descriptiva. Con un nombre de dominio personalizado regional, puede asignar etapas de la API de HTTP y REST al mismo nombre de dominio personalizado y usar la autenticación TLS mutua.

Consideraciones

A continuación, se muestran consideraciones para el nombre de dominio personalizado regional:

- Debe proporcionar un certificado de ACM específico de la región. Este certificado debe estar en la misma región que la API. Para obtener más información sobre cómo crear o cargar un certificado de nombre de dominio personalizado, consulte the section called “Preparación de certificados en AWS Gestor de certificados”.
- Cuando crea un nombre de dominio personalizado regional (o migra uno) con un certificado de ACM, API Gateway crea un rol vinculado al servicio en su cuenta. El rol vinculado al servicio es necesario para asociar el certificado de ACM a su punto de conexión regional. El rol se denomina AWSServiceRoleForAPIGateway y llevará asociada la política administrada APIGatewayServiceRolePolicy. Para obtener más información acerca de cómo usar el rol vinculado al servicio, consulte Uso de roles vinculados a servicios.
- Después de crear el nombre de dominio personalizado regional, deberá crear un registro DNS para que el nombre de dominio personalizado apunte al nombre de dominio regional. Esto permite que el tráfico que se dirige al nombre de dominio personalizado se enrute al nombre de host regional de la API.

  El registro DNS puede ser el registro CNAME o un registro de alias A. Si utiliza Route 53 como proveedor de DNS, cree un registro de alias A. Si utiliza un proveedor de DNS externo, utilice un registro CNAME. Si utiliza un registro CNAME y crea un punto de conexión de VPC de interfaz de

API Gateway con el DNS privado habilitado para una API privada, no podrá resolver el nombre de dominio personalizado en la VPC que aloja la API privada.

Creación de un nombre de dominio personalizado regional

En el siguiente procedimiento se muestra cómo crear un nombre de dominio personalizado regional. Una vez completado este procedimiento, cree una regla de enrutamiento para enrutar las etapas de la API al nombre de dominio personalizado.

Consola de administración de AWS

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija Custom domain names (Nombres de dominios personalizados) en el panel de navegación principal.
3. Seleccione Create (Crear).
4. En Nombre de dominio, escriba un nombre de dominio.
5. En Modo de enrutamiento, elija Solo reglas de enrutamiento.

   En este modo de enrutamiento, solo puede enviar tráfico desde el nombre de dominio personalizado a las API mediante reglas de enrutamiento. Para obtener más información, consulte the section called “Envío de tráfico a las API a través del nombre de dominio personalizado en API Gateway”.
6. En Versión mínima de TLS, seleccione una versión.
7. En Configuración de punto de conexión, para Tipo de punto de conexión de la API, elija Regional.
8. Elija un certificado de ACM. El certificado debe estar en la misma región que la API.
9. Seleccione Create (Crear).

AWS CLI

El siguiente comando create-domain-name permite crear un nombre de dominio personalizado:

```
aws apigatewayv2 create-domain-name \
    --domain-name 'regional.example.com' \
    --domain-name-configurations CertificateArn=arn:aws:acm:us-
west-2:123456789012:certificate/123456789012-1234-1234-1234-12345678 \
    --routing-mode ROUTING_RULE_ONLY
```

El resultado será similar al siguiente:

```
{
    "ApiMappingSelectionExpression": "$request.basepath",
    "DomainName": "regional.example.com",
    "DomainNameConfigurations": [
        {
            "ApiGatewayDomainName": "d-numh1z56v6.execute-api.us-
west-2.amazonaws.com",
            "CertificateArn": "arn:aws:acm:us-
west-2:123456789012:certificate/123456789012-1234-1234-1234-12345678",
            "DomainNameStatus": "AVAILABLE",
            "EndpointType": "REGIONAL",
            "HostedZoneId": "Z2OJLYMUO9EFXC",
            "SecurityPolicy": "TLS_1_2"
        }
        "RoutingMode": "ROUTING_RULE_ONLY"
    ]
}
```

El valor de propiedad `DomainNameConfigurations` devuelve el nombre de host de la API regional. Debe crear un registro DNS para apuntar su nombre de dominio personalizado a este nombre de dominio regional. Esto permite que el tráfico que se dirige al nombre de dominio personalizado se enrute al nombre de host de esta API regional.

Creación de una regla de enrutamiento para el nombre de dominio personalizado regional

Después de crear el nombre de dominio personalizado, configure cómo se enruta el tráfico desde el nombre de dominio personalizado a las API. Dado que establece el modo de enrutamiento en `ROUTING_RULE_ONLY`, utiliza reglas de enrutamiento para enrutar las solicitudes entrantes al nombre de dominio personalizado hacia las API.

En este ejemplo, crea una regla catch-all que enruta todas las solicitudes entrantes al nombre de dominio personalizado a una etapa de la API. También puede configurar reglas de enrutamiento basadas en diferentes condiciones de encabezado y ruta. Para obtener más información, consulte the section called “Reglas de enrutamiento”.

Consola de administración de AWS

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.

2. Elija un nombre de dominio personalizado.
3. En la pestaña Detalles de enrutamiento, elija Agregar regla de enrutamiento.
4. Elija Agregar una nueva condición para agregar una nueva condición.
5. Mantenga esta regla sin condiciones. Esto enruta todas las solicitudes al nombre de dominio personalizado a la API de destino y etapa de destino.
6. En Acción, utilice la lista desplegable para seleccionar la API de destino y la etapa de destino.
7. Elija Siguiente.
8. En el campo de prioridad, introduzca **`100`**.

   API Gateway evalúa las reglas en orden de prioridad, del valor más bajo al más alto. Dado que se trata de una regla catch-all, utilice una prioridad alta para que API Gateway pueda realizar la coincidencia primero con cualquier regla adicional que cree.
9. Elija Crear regla de enrutamiento.

AWS CLI

El siguiente comando `create-routing-rule` crea una regla de enrutamiento catch-all:

```
aws apigatewayv2 create-routing-rule \
  --domain-name 'regional.example.com' \
  --priority 100 \
  --conditions  \
  --actions '[{
    "InvokeApi": {
      "ApiId": "a1b2c3",
      "Stage": "prod"
    }
  }]'
```

Puede cambiar el modo de enrutamiento y crear nuevas reglas en cualquier momento. Para obtener más información, consulte the section called “Envío de tráfico a las API a través del nombre de dominio personalizado en API Gateway”.

Creación de un registro de DNS para el nombre de dominio personalizado regional

Después de crear el nombre de dominio personalizado y las asignaciones de rutas base, debe crear un registro DNS para que el nombre de dominio personalizado apunte al nombre de dominio regional que se acaba de crear.

Consola de administración de AWS

Para usar la Consola de administración de AWS, siga la documentación de Route 53 sobre la configuración de Route 53 para enrutar el tráfico a API Gateway.

AWS CLI

Para configurar sus registros de DNS para asignar el nombre de dominio regional personalizado al nombre de host del ID de zona alojada designado, cree primero un archivo JSON que contenga la configuración a fin de establecer un registro de DNS para el nombre de dominio regional.

En el siguiente `setup-dns-record.json` se muestra cómo crear un registro `A` de DNS para asignar un nombre de dominio personalizado regional (`regional.example.com`) a su nombre de host regional (`d-numh1z56v6.execute-api.us-west-2.amazonaws.com`) aprovisionado como parte de la creación de nombres de dominio personalizados. Las propiedades `DNSName` y `HostedZoneId` de `AliasTarget` pueden tener los valores `regionalDomainName` y `regionalHostedZoneId`, respectivamente, del nombre de dominio personalizado. También puede obtener los ID de zona alojada regionales de Route 53 en Puntos de conexión y cuotas de Amazon API Gateway.

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

El siguiente comando change-resource-record-sets permite crear un registro de DNS para el nombre de dominio personalizado regional:

```
aws route53 change-resource-record-sets \
    --hosted-zone-id Z2OJLYMUO9EFXC \
    --change-batch file://path/to/your/setup-dns-record.json
```

Reemplace `hosted-zone-id` por el ID de zona alojada de Route 53 del conjunto de registros de DNS de la cuenta. El valor del parámetro `change-batch` apunta a un archivo JSON (*`setup-dns-record.json`*) de una carpeta (*`ruta/a/su`*).

Configuración de un nombre de dominio personalizado optimizado para la periferia para en API Gateway

Cuando crea un nombre de dominio personalizado para una API optimizada para la periferia, API Gateway configura una distribución de Amazon CloudFront y un registro de DNS para asignar el nombre de dominio de API al nombre de dominio de distribución de CloudFront. Las solicitudes para la API se redirigen a API Gateway a través de la distribución de CloudFront asignada. Este mapeo es para solicitudes de API que estén enlazadas para que el nombre de dominio personalizado se dirija a API Gateway a través de la distribución de CloudFront asignada.

Consideraciones

A continuación, se muestran las siguientes consideraciones para el nombre de dominio personalizado optimizado para la periferia:

- Para configurar un nombre de dominio personalizado optimizado para bordes o para actualizar su certificado, debe tener permiso para actualizar las distribuciones de CloudFront.

  Se requieren los siguientes permisos para actualizar las distribuciones de CloudFront:

  JSON

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

```
                "cloudfront:updateDistribution"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

- Debe solicitar o importar un certificado para su nombre de dominio personalizado optimizado para la periferia en la región Este de EE. UU. (Norte de Virginia) - `us-east-1`.
- La distribución de CloudFront que ha creado API Gateway es propiedad de un cuenta específica de la región afiliada a API Gateway. Cuando realice un seguimiento de las operaciones para crear y actualizar esta distribución de CloudFront en CloudTrail, deberá utilizar este ID de cuenta de API Gateway. Para obtener más información, consulte Registro de la creación del nombre de dominio personalizado en CloudTrail.
- API Gateway admite nombres de dominio personalizados optimizados para bordes mediante la indicación de nombre de servidor (SNI) en la distribución de CloudFront. Para obtener más información acerca de cómo usar nombres de dominio personalizados en una distribución de CloudFront, incluido el formato de certificado necesario y el tamaño máximo de una longitud de clave de certificado, consulte Uso de nombres de dominio alternativos y HTTPS en la Guía para desarrolladores de Amazon CloudFront.
- Un nombre de dominio personalizado optimizado para la periferia tarda unos 40 minutos en estar listo.
- Una vez creado su nombre de dominio personalizado optimizado para la periferia, debe crear un registro de DNS para asignar el nombre de dominio personalizado al nombre de distribución de CloudFront.

Creación de un nombre de dominio personalizado optimizado para la periferia

El siguiente procedimiento describe cómo crear un nombre de dominio personalizado optimizado para la periferia para una API.

Consola de administración de AWS

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija Custom domain names (Nombres de dominios personalizados) en el panel de navegación principal.

3. Elija Agregar nombre de dominio.
4. En Nombre de dominio, escriba un nombre de dominio.
5. En Modo de enrutamiento, elija API_MAPPING_ONLY.
6. En Tipo de punto de conexión de la API, elija Optimizado para periferia.
7. Elija una versión mínima de TLS.
8. Elija un certificado de ACM.
9. Elija Agregar nombre de dominio.

REST API

1. Llame a domainname:create, especificando el nombre de dominio personalizado y el ARN de un certificado almacenado en AWS Gestor de certificados.

   Si la llamada a la API se realiza correctamente, se devuelve una respuesta `201 Created`, la cual contiene el ARN del certificado, así como el nombre de distribución de CloudFront asociado en su carga.
2. Anote el nombre de dominio de distribución de CloudFront que aparece en el resultado. Lo necesitará en el siguiente paso para establecer el destino de alias de registro A del dominio personalizado en el DNS.

Para ver ejemplos de código de esta llamada a la API de REST, consulte domainname:create.

Un nombre de dominio personalizado optimizado para la periferia tarda unos 40 minutos en estar listo, pero la consola mostrará inmediatamente el nombre de dominio de distribución de CloudFront asociado, con el formato *`distribution-id`*`.cloudfront.net`, junto con el ARN del certificado. Mientras tanto, puede crear una asignación de ruta base o una regla de enrutamiento y, a continuación, configurar el alias de registro de DNS para asignar el nombre de dominio personalizado al nombre de dominio de distribución de CloudFront asociado.

Configurar el mapeo de ruta base de una API con un nombre de dominio personalizado como su nombre de host

Debido a que ha establecido el modo de enrutamiento en `API_MAPPING_ONLY`, puede usar la asignación de ruta base para utilizar un único nombre de dominio personalizado como el nombre de host de varias API. De este modo, una API estará accesible a través de la combinación del nombre de dominio personalizado y la ruta base asociada.

Por ejemplo, si en API Gateway creó una API llamada `PetStore` y otra API con el nombre `Dogs`, y configuró un nombre de dominio personalizado de `api.example.com`, puede establecer la URL de la API `PetStore` como `https://api.example.com`.

Esto asocia la API `PetStore` con la ruta base de una cadena vacía. Si establece la URL de la API `PetStore` como `https://api.example.com/PetStore`, se asocia la API `PetStore` a la ruta base de `PetStore`. Puede asignar una ruta base de `MyDogList` para la API `Dogs`. La dirección URL de `https://api.example.com/MyDogList` es entonces la URL raíz de la API `Dogs`.

Para configurar las asignaciones de API en varios niveles, solo puede usar un nombre de dominio personalizado regional. No se admiten los nombres de dominio personalizados optimizados para la periferia. Para obtener más información, consulte the section called “Mapeos de la API”.

El siguiente procedimiento configura asignaciones de API para asignar rutas desde el nombre de dominio personalizado a las etapas de API.

Consola de administración de AWS

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Elija Custom domain names (Nombres de dominio personalizados) en el panel de navegación principal de la consola de API Gateway.
3. Elija un nombre de dominio personalizado.
4. Elija Configure API mappings (Configurar asignaciones de API).
5. Seleccione Add new mapping (Agregar nueva asignación).
6. Especifique la API, Stage (Etapa) y Route (Ruta) (opcional) para la asignación.
7. Seleccione Save.

REST API

Llame a basepathmapping:create en un nombre de dominio personalizado concreto especificando los elementos `basePath`, `restApiId` y una propiedad `stage` de implementación en la carga de la solicitud.

La llamada a la API, si se realiza correctamente, devuelve una respuesta `201 Created`.

Para ver ejemplos de código de la llamada a la API de REST, consulte basepathmapping:create.

Para obtener mayor flexibilidad sobre cómo enrutar el tráfico a las API, puede cambiar el modo de enrutamiento a `ROUTING_RULE_ONLY` o `ROUTING_RULE_THEN_API_MAPPING` y crear una regla de enrutamiento. Para obtener más información, consulte the section called “Envío de tráfico a las API a través del nombre de dominio personalizado en API Gateway”.

Creación de un registro de DNS para el nombre de dominio personalizado optimizado para la periferia

Después de iniciar la creación de su nombre de dominio personalizado optimizado para la periferia, configure el alias de registro de DNS.

Le recomendamos que use Route 53 para crear un alias de registro A para su nombre de dominio personalizado y especificar el nombre de dominio de distribución de CloudFront como el alias de destino. Esto significa que Route 53 pueden enrutar su nombre de dominio personalizado aunque se trate de un ápex de zona. Para obtener más información, consulte Elección entre registros de alias y sin alias en la guía para desarrolladores de Amazon Route 53.

Para obtener instrucciones sobre Amazon Route 53, consulte Routing traffic to an Amazon API Gateway API by using your domain name en la Guía para desarrolladores de Amazon Route 53.

Registro de la creación del nombre de dominio personalizado en CloudTrail

Cuando CloudTrail está habilitado para registrar llamadas de API Gateway realizadas por su cuenta, API Gateway registra las actualizaciones de la distribución de CloudFront asociadas cuando se crea o se actualiza un nombre de dominio personalizado para una API. Estos registros están disponibles en `us-east-1`. Como estas distribuciones de CloudFront son propiedad de API Gateway, cada una de estas distribuciones de CloudFront registradas se identifican mediante uno de los siguientes ID de cuenta de API Gateway específicos de cada región, en lugar de mediante el ID de la cuenta del propietario de la API.

Región	ID de cuenta
us-east-1	392220576650
us-east-2	718770453195
us-west-1	968246515281
us-west-2	109351309407

Región	ID de cuenta
ca-central-1	796887884028
eu-west-1	631144002099
eu-west-2	544388816663
eu-west-3	061510835048
eu-central-1	474240146802
eu-central-2	166639821150
eu-north-1	394634713161
eu-south-1	753362059629
eu-south-2	359345898052
ap-northeast-1	969236854626
ap-northeast-2	020402002396
ap-northeast-3	360671645888
ap-southeast-1	195145609632
ap-southeast-2	798376113853
ap-southeast-3	652364314486
ap-southeast-4	849137399833
ap-south-1	507069717855
ap-south-2	644042651268
ap-east-1	174803364771
sa-east-1	287228555773

Región	ID de cuenta
me-south-1	855739686837
me-central-1	614065512851

Rotación de un certificado importado en ACM

ACM se encarga automáticamente de renovar los certificados que emite. No necesita rotar ningún certificado emitido por ACM para los nombres de dominio personalizados. CloudFront lo gestiona por usted.

No obstante, si importa un certificado en ACM y lo usa para un nombre de dominio personalizado, debe rotar el certificado antes de que venza. Esto implica importar un nuevo certificado de terceros para el nombre de dominio y reemplazar el certificado existente por el nuevo. Necesita repetir el proceso cuando el certificado que acaba de importar venza. De forma alternativa, puede solicitar a ACM que emita un nuevo certificado para el nombre de dominio y reemplazar el existente por el nuevo certificado emitido por ACM. A partir de entonces, puede dejar que ACM y CloudFront se ocupen de administrar la rotación de certificados automáticamente. Para crear o importar un nuevo certificado de ACM, siga los pasos de the section called “Creación o importación de un certificado SSL/TLS en ACM”.

En el siguiente procedimiento se describe cómo rotar un certificado para un nombre de dominio.

> **Note**
>
> Se tarda unos 40 minutos en rotar un certificado importado a ACM.

Consola de administración de AWS

1. Solicite o importe un certificado en ACM.
2. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
3. Elija Custom domain names (Nombres de dominio personalizados) en el panel de navegación principal de la consola de API Gateway.
4. Elija un nombre de dominio personalizado optimizado para la periferia.
5. En Configuración de punto de conexión, elija Editar.

6. En Certificado de ACM, seleccione un certificado en la lista desplegable.
7. Elija Guardar cambios para empezar a rotar el certificado para el nombre de dominio personalizado.

REST API

Llame a la acción domainname:update especificando el ARN del nuevo certificado de ACM para el nombre de dominio especificado.

AWS CLI

El siguiente update-domain-name actualiza el certificado de ACM de un nombre de dominio optimizado para bordes.

```
aws apigateway update-domain-name \
    --domain-name edge.example.com \
    --patch-operations "op='replace',path='/certificateArn',value='arn:aws:acm:us-
east-2:111122223333:certificate/CERTEXAMPLE123EXAMPLE'"
```

El siguiente update-domain-name actualiza el certificado de ACM de un nombre de dominio regional.

```
aws apigateway update-domain-name \
    --domain-name regional.example.com \
    --patch-operations "op='replace',path='/
regionalCertificateArn',value='arn:aws:acm:us-east-2:111122223333:certificate/
CERTEXAMPLE123EXAMPLE'"
```

Llamada a la API con nombres de dominio personalizados cuando utilice una asignación de ruta base

Llamar a una API con un nombre de dominio personalizado es lo mismo que llamar a la API con su nombre de dominio predeterminado, siempre que se use la URL correcta.

En los siguientes ejemplos se compara y se contrasta un conjunto de direcciones URL predeterminadas y las URL personalizadas correspondientes de dos API (`udxjef` y `qf3duz`) en una región especificada (`us-east-1`) y de un nombre de dominio personalizado (`api.example.com`) especificado.

ID de API	Etapa	URL predeterminada	Ruta base	URL personalizada
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.amazonaws.com/tst	/petdepot	https://api.example.com/petdepot
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

Para obtener mayor flexibilidad sobre cómo enrutar el tráfico a las API, puede crear una regla de enrutamiento. Para obtener más información, consulte the section called “Envío de tráfico a las API a través del nombre de dominio personalizado en API Gateway”.

API Gateway admite nombres de dominio personalizados para una API mediante el uso de la indicación de nombre de servidor (SNI). Puede invocar la API con un nombre de dominio personalizado mediante un navegador o una biblioteca cliente que admita SNI.

API Gateway aplica SNI en la distribución CloudFront. Para obtener información sobre cómo CloudFront utiliza nombres de dominio personalizados, consulte Certificados SSL personalizados de Amazon CloudFront.

Migración de un nombre de dominio personalizado a otro tipo punto de conexión de API en API Gateway

Puede migrar el nombre de dominio personalizado entre puntos de conexión optimizados para bordes y regionales. No es posible migrar un nombre de dominio personalizado público a un nombre de dominio personalizado privado. En primer lugar, debe añadir el nuevo tipo de configuración de punto de conexión a la lista `endpointConfiguration.types` existente para el nombre de dominio personalizado. A continuación, debe configurar un registro DNS para que el nombre de dominio personalizado señale al punto de conexión recién aprovisionado. Por último, debe quitar el punto de conexión de los nombres de dominio personalizados obsoletos.

Consideraciones

A continuación, se ofrecen algunas consideraciones para migrar el dominio personalizado entre un punto de conexión de API regional y un punto de conexión de API optimizado para la periferia:

- Un nombre de dominio personalizado optimizado para la periferia requiere un certificado proporcionado por ACM de la región Este de EE. UU. (Norte de Virginia) - `us-east-1`. Este certificado se distribuye a todas las ubicaciones geográficas.
- Un nombre de dominio personalizado regional requiere un certificado proporcionado por ACM en la misma región que aloja la API. Puede migrar un nombre de dominio personalizado optimizado para la periferia que no se encuentra en la región `us-east-1` a un nombre de dominio personalizado regional mediante la solicitud de un nuevo certificado de ACM desde la región local para la API.
- Puede tardar hasta 60 segundos en completarse una migración entre un nombre de dominio personalizado optimizado para la periferia y un nombre de dominio personalizado regional. El tiempo de migración también depende de cuándo se actualicen los registros de DNS.
- Solo puede agregar una configuración de punto de conexión adicional si el modo de acceso de punto de conexión está configurado en `BASIC`. Una vez que tenga dos configuraciones de punto de conexión, no podrá cambiar el modo de acceso de punto de conexión. Para obtener más información, consulte the section called “Modo de acceso de punto de conexión”.
- Si el nombre de dominio personalizado usa una política de seguridad que comienza con `SecurityPolicy_`, cuando agrega un nuevo tipo de configuración de punto de conexión, el modo de acceso al punto de conexión es el mismo en ambos tipos de punto de conexión y debe

elegir una política de seguridad que comience con `SecurityPolicy_` para el nuevo tipo de configuración de punto de conexión.

Migración de nombres de dominio personalizados

> **Note**
>
> Para completar la migración, asegúrese de eliminar el punto de conexión obsoleto del nombre de dominio personalizado.

En el siguiente procedimiento se muestra cómo migrar un nombre de dominio personalizado optimizado para la periferia a un nombre de dominio personalizado regional.

Consola de administración de AWS

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija Custom domain names (Nombres de dominios personalizados) en el panel de navegación principal.
3. Elija un nombre de dominio personalizado optimizado para la periferia.
4. En Configuración de punto de conexión, elija Editar.
5. Elija Agregar punto de conexión regional.
6. En Certificado de ACM, elija un certificado.

   El certificado regional debe estar en la misma región que la API regional.
7. Seleccione Save changes (Guardar cambios).
8. Configure un registro de DNS para que el nombre de dominio personalizado regional apunte a este nombre de host regional. Para obtener más información, consulte Configuración de Route 53 para enrutar el tráfico a API Gateway.
9. Después de confirmar que la configuración de DNS utiliza el punto de conexión correcto, elimine la configuración de punto de conexión optimizado para la periferia. Elija el nombre del dominio personalizado y, después, elija Eliminar para Configuración de punto de conexión (optimizado para bordes).
10. Confirme su elección y elimine el punto de conexión.

AWS CLI

El siguiente comando update-domain-name permite migrar el nombre de dominio personalizado optimizado para bordes a un nombre de dominio personalizado regional:

```
aws apigateway update-domain-name \
    --domain-name 'api.example.com' \
    --patch-operations  '[
        { "op":"add", "path": "/endpointConfiguration/types","value": "REGIONAL" },
        { "op":"add", "path": "/regionalCertificateArn", "value": "arn:aws:acm:us-
west-2:123456789012:certificate/cd833b28-58d2-407e-83e9-dce3fd852149" }
      ]'
```

El certificado regional debe corresponder a la misma región que la API regional.

El resultado será similar al siguiente:

```
{
    "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"
}
```

En el caso del nombre de dominio personalizado regional actualizado, la propiedad resultante `regionalDomainName` devuelve el nombre de host de la API regional. Debe configurar un registro DNS para que el nombre de dominio personalizado regional señale a este nombre de host regional. Esto permite que el tráfico que se dirige al nombre de dominio personalizado sea redirigido al host regional.

Una vez configurado el registro de DNS, puede quitar el nombre de dominio personalizado optimizado para la periferia. El siguiente comando update-domain-name permite eliminar el nombre de dominio personalizado optimizado para bordes.

```
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"}
        ]'
```

En el siguiente procedimiento se muestra cómo migrar un nombre de dominio personalizado optimizado para bordes que utilice una política de seguridad mejorada a un nombre de dominio personalizado regional que también utilice una política de seguridad mejorada.

Consola de administración de AWS

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija Custom domain names (Nombres de dominios personalizados) en el panel de navegación principal.
3. Elija un nombre de dominio personalizado optimizado para la periferia.
4. En Configuración de punto de conexión, elija Editar.
5. Elija Agregar punto de conexión regional.
6. En Certificado de ACM, elija un certificado.

   El certificado regional debe estar en la misma región que la API regional.
7. Para Política de seguridad, elija una política de seguridad que comience con `SecurityPolicy_`.
8. Para Modo de acceso de punto de conexión, elija Básico.
9. Seleccione Save changes (Guardar cambios).
10. Configure un registro de DNS para que el nombre de dominio personalizado regional apunte a este nombre de host regional. Para obtener más información, consulte Configuración de Route 53 para enrutar el tráfico a API Gateway.
11. Después de confirmar que la configuración de DNS utiliza el punto de conexión correcto, elimine la configuración de punto de conexión optimizado para la periferia. Elija el nombre del

dominio personalizado y, después, elija Eliminar para Configuración de punto de conexión (optimizado para bordes).

12. Confirme su elección y elimine el punto de conexión.

AWS CLI

El siguiente comando update-domain-name permite migrar el nombre de dominio personalizado optimizado para bordes a un nombre de dominio personalizado regional:

```
aws apigateway update-domain-name \
    --domain-name 'api.example.com' \
    --patch-operations  '[
        { "op":"add", "path": "/endpointConfiguration/types","value": "REGIONAL" },
        { "op":"replace", "path": "/securityPolicy",
 "value":"SecurityPolicy_TLS13_1_3_2025_09"},
        { "op":"add", "path": "/regionalCertificateArn", "value": "arn:aws:acm:us-
west-2:123456789012:certificate/cd833b28-58d2-407e-83e9-dce3fd852149" }
      ]'
```

El certificado regional debe corresponder a la misma región que la API regional.

El resultado será similar al siguiente:

```
{
    "certificateArn": "arn:aws:acm:us-
east-1:123456789012:certificate/34a95aa1-77fa-427c-aa07-3a88bd9f3c0a",
    "certificateName": "edge-cert",
    "certificateUploadDate": "2017-10-16T23:22:57Z",
    "distributionDomainName": "d1frvgze7vy1bf.cloudfront.net",
    "domainName": "api.example.com",
    "endpointConfiguration": {
        "types": [
            "EDGE",
            "REGIONAL"
        ]
    },
    "securityPolicy": "SecurityPolicy_TLS13_1_3_2025_09",
    "endpointAccessMode": "BASIC",
    "regionalCertificateArn": "arn:aws:acm:us-west-2:123456789012:certificate/
cd833b28-58d2-407e-83e9-dce3fd852149",
    "regionalDomainName": "d-fdisjghyn6.execute-api.us-west-2.amazonaws.com"
```

```
}
```

En el caso del nombre de dominio personalizado regional actualizado, la propiedad resultante `regionalDomainName` devuelve el nombre de host de la API regional. Debe configurar un registro DNS para que el nombre de dominio personalizado regional señale a este nombre de host regional. Esto permite que el tráfico que se dirige al nombre de dominio personalizado sea redirigido al host regional.

Una vez configurado el registro de DNS, puede quitar el nombre de dominio personalizado optimizado para la periferia. El siguiente comando update-domain-name permite eliminar el nombre de dominio personalizado optimizado para bordes.

```
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"}
        ]'
```

En el siguiente procedimiento se muestra cómo migrar un nombre de dominio personalizado regional a un nombre de dominio personalizado optimizado para la periferia.

Consola de administración de AWS

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Nombres de dominio personalizados.
3. Elija un nombre de dominio personalizado regional.
4. En Configuración de punto de conexión, elija Editar.
5. Elija Agregar punto de conexión optimizado para la periferia.
6. En Certificado de ACM, elija un certificado.

   El certificado del dominio optimizado para sistemas perimetrales debe crearse en la región `us-east-1`.
7. Seleccione Save.

8. Configure un registro de DNS para que el nombre de dominio personalizado optimizado para la periferia apunte a este nombre de host optimizado para la periferia. Para obtener más información, consulte Configuración de Route 53 para enrutar el tráfico a API Gateway.
9. Después de confirmar que la configuración de DNS utiliza el punto de conexión correcto, elimine la configuración de punto de conexión regional. Elija el nombre del dominio personalizado y, después, elija Eliminar para Configuración de punto de conexión (regional).
10. Confirme su elección y elimine el punto de conexión.

AWS CLI

El siguiente comando update-domain-name permite migrar el nombre de dominio personalizado regional a un nombre de dominio personalizado optimizado para bordes:

```
aws apigateway update-domain-name \
    --domain-name 'api.example.com' \
    --patch-operations  '[
        { "op":"add", "path": "/endpointConfiguration/types","value": "EDGE" },
        { "op":"add", "path": "/certificateName", "value": "edge-cert" },
 {"op":"add", "path": "/certificateArn", "value": "arn:aws:acm:us-
east-1:738575810317:certificate/34a95aa1-77fa-427c-aa07-3a88bd9f3c0a"}
      ]'
```

El certificado del dominio optimizado para sistemas perimetrales debe crearse en la región `us-east-1`.

El resultado será similar al siguiente:

```
{
    "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"
}
```

Para el nombre de dominio personalizado especificado, API Gateway devuelve el nombre de host de la API optimizada para bordes como el valor de la propiedad `distributionDomainName`. Debe configurar un registro DNS para que el nombre de dominio personalizado optimizado para sistemas perimetrales señale a este nombre de dominio de distribución. Esto permite que el tráfico que se dirige al nombre de dominio personalizado optimizado para sistemas perimetrales sea redirigido al nombre de host de la API optimizada para límites.

Una vez configurado el registro de DNS, puede quitar el tipo de punto de conexión `REGION` del nombre de dominio personalizado. El siguiente comando update-domain-name permite eliminar el tipo de punto de conexión regional:

```
aws apigateway update-domain-name \
    --domain-name api.example.com \
    --patch-operations '[
        {"op":"remove", "path":"/endpointConfiguration/types", value:"REGIONAL"},
        {"op":"remove", "path":"regionalCertificateArn"}
      ]'
```

El resultado es similar al siguiente:

```
{
    "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"
    }
}
```

Envío de tráfico a las API a través del nombre de dominio personalizado en API Gateway

Cuando configura el modo de enrutamiento para el nombre de dominio personalizado, establece cómo se dirige el tráfico entrante a las API. Puede enviar tráfico a las API mediante reglas de enrutamiento, asignaciones de API o reglas de enrutamiento y asignaciones de API. En la siguiente sección se explica cuándo utilizar reglas de enrutamiento, cuándo utilizar asignaciones de API y cómo establecer el modo de enrutamiento para el nombre de dominio personalizado.

Cuándo utilizar reglas de enrutamiento

Cuando utiliza reglas de enrutamiento, dirige las solicitudes entrantes que coinciden con determinadas condiciones a etapas específicas de las API de REST. Por ejemplo, una regla puede enrutar una solicitud a la etapa `production` de la API de REST `users` si contiene el encabezado `version:v1` y la ruta base `/users`. Utilice reglas de enrutamiento para crear topologías avanzadas de enrutamiento dinámico que admitan casos de uso como las pruebas A/B o el aumento del uso de nuevas versiones de las API.

Le recomendamos que, cuando dirija el tráfico a una API de REST, utilice reglas de enrutamiento para el nombre de dominio personalizado. Puede recrear cualquier asignación de API mediante reglas de enrutamiento. Para obtener más información, consulte the section called “Nueva creación de una asignación de API mediante reglas de enrutamiento”.

En el caso de las API de REST, también puede utilizar conjuntamente reglas de enrutamiento y asignaciones de API. Cuando utiliza reglas de enrutamiento y asignaciones de API de forma conjunta, API Gateway siempre evalúa las reglas de enrutamiento antes de evaluar cualquier asignación de API. Utilice las reglas de enrutamiento y las asignaciones de API conjuntamente para migrar los nombres de dominio personalizados actuales o para explorar las reglas de enrutamiento.

Consideraciones sobre las reglas de enrutamiento

Las siguientes consideraciones pueden afectar la utilización de las reglas de enrutamiento:

- Las API de WebSocket o HTTP no son compatibles como API de destino para las reglas de enrutamiento.
- Si el nombre de dominio personalizado tiene asignaciones de API tanto a las API de REST como de HTTP, no se admiten las reglas de enrutamiento.

- Puede crear una regla de enrutamiento para un dominio personalizado privado a una API de REST privada. Puede crear una regla de enrutamiento para un dominio personalizado público a una API regional u optimizada para la periferia.
- No puede crear una regla de enrutamiento para un dominio personalizado público a una API privada. No puede crear una regla de enrutamiento para un dominio personalizado privado a una API pública.

Elección entre reglas de enrutamiento y asignaciones de API

Le recomendamos que, siempre que sea posible, utilice reglas de enrutamiento. Solo utilice asignaciones de API para enviar tráfico a una API de HTTP o WebSocket.

Establecimiento del modo de enrutamiento para el nombre de dominio personalizado

Puede elegir qué modo de enrutamiento utiliza API Gateway para enrutar el tráfico a las API. Para obtener más información, consulte the section called “Envío de tráfico a las API a través del nombre de dominio personalizado en API Gateway”. En esta sección se explican los modos de enrutamiento para nombres de dominio personalizados. Debe establecer un modo de enrutamiento para el nombre de dominio personalizado a fin de enrutar el tráfico a las API. Se admiten los siguientes modos de enrutamiento:

- ROUTING_RULE_THEN_API_MAPPING: utilice este modo para enviar tráfico a las API tanto con reglas de enrutamiento como con asignaciones de API. En este modo, todas las reglas de enrutamiento tienen prioridad sobre cualquier asignación de API. Para ver un ejemplo de este modo, consulte the section called “Ejemplo 2: reglas de enrutamiento y asignaciones de API”.
- ROUTING_RULE_ONLY: utilice este modo para permitir solo que las reglas de enrutamiento envíen tráfico a las API. Cuando el dominio personalizado utiliza este modo, no puede crear una asignación de API, pero puede utilizar el comando get-api-mappings para verlas. Los intermediarios que llaman a la API no pueden utilizar asignaciones de API para acceder a este nombre de dominio.
- API_MAPPING_ONLY: utilice este modo para permitir solo que las asignaciones de API envíen tráfico a las API. Cuando el nombre de dominio personalizado utiliza este modo, no puede crear una regla de enrutamiento, pero puede utilizar el comando `list-routing-rules` para verlas. Los intermediarios que llaman a la API no pueden utilizar reglas de enrutamiento para acceder a este nombre de dominio.

  Este es el modo de enrutamiento predeterminado para todos los nombres de dominio existentes y para cualquier nombre de dominio nuevo que cree.

Cuando crea un nombre de dominio personalizado mediante `apigateway`, `API_MAPPING_ONLY` se llama `BASE_PATH_MAPPING_ONLY` y `ROUTING_RULE_THEN_API_MAPPING` se llama `ROUTING_RULE_THEN_BASE_PATH_MAPPING`. Este comportamiento solo está presente para la AWS CLI, CloudFormation o cualquier SDK, no en la Consola de administración de AWS.

El procedimiento siguiente muestra cómo cambiar el modo de enrutamiento de un nombre de dominio personalizado existente. Cuando cambie el modo de enrutamiento del nombre de dominio personalizado, los intermediarios que llamen a la API no podrán acceder al nombre de dominio con ningún modo de enrutamiento no admitido.

Consola de administración de AWS

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija Custom domain names (Nombres de dominios personalizados) en el panel de navegación principal.
3. Elija un nombre de dominio personalizado.
4. En Detalles del dominio, elija Editar.
5. En Modo de enrutamiento, elija ROUTING_RULE_THEN_API_MAPPING.
6. Seleccione Save.

Si cambia el modo de enrutamiento a `ROUTING_RULE_ONLY` o `API_MAPPING_ONLY`, cualquier asignación de API o regla de enrutamiento que haya creado se eliminará de la página de detalles del nombre de dominio de la consola. Si cambia el modo de enrutamiento para que admita reglas de enrutamiento o asignaciones de API, estos recursos volverán a aparecer.

AWS CLI - apigatewayv2

El siguiente comando update-domain-name actualiza un nombre de dominio para utilizar el modo de enrutamiento `ROUTING_RULE_THEN_API_MAPPING`:

```
aws apigatewayv2 update-domain-name \
  --domain-name 'api.example.com' \
  --routing-mode "ROUTING_RULE_THEN_API_MAPPING"
```

El resultado será similar al siguiente:

```
{
"ApiMappingSelectionExpression": "$request.basepath",
"DomainName": "api.example.com",
```

```
"DomainNameArn": "arn:aws:apigateway:us-west-2::/domainnames/api.example.com",
"DomainNameConfigurations": [
  {
      "ApiGatewayDomainName": "d-abcdefg.execute-api.us-west-2.amazonaws.com",
      "CertificateArn": "arn:aws:acm:us-west-2:111122223333:certificate/
abcdefg-123456-abcdefg",
      "DomainNameStatus": "AVAILABLE",
      "EndpointType": "REGIONAL",
      "HostedZoneId": "Z2OJLYMUO9EFXC",
      "SecurityPolicy": "TLS_1_2"
   }
 ],
"RoutingMode": "ROUTING_RULE_THEN_API_MAPPING",
"Tags": {}
}
```

AWS CLI - apigateway

El siguiente comando update-domain-name actualiza un nombre de dominio personalizado privado para utilizar el modo de enrutamiento ROUTING_RULE_THEN_BASE_PATH_MAPPING:

```
aws apigateway update-domain-name \
  --domain-name 'private.example.com' \
  --patch-operations "op='replace',path='/
routingMode',value='ROUTING_RULE_THEN_BASE_PATH_MAPPING'"
```

El resultado será similar al siguiente:

```
{
"domainName": "private.example.com",
"domainNameId": "abcd1234",
"domainNameArn": "arn:aws:apigateway:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234",
"certificateArn": "arn:aws:acm:us-west-2:111122223333:certificate/
a1b2c3d4-5678-90ab-cdef",
"certificateUploadDate": "2024-09-10T10:31:20-07:00",
"endpointConfiguration": {
  "types": [
    "PRIVATE"
   ],
  "ipAddressType": "dualstack"
  },
"domainNameStatus": "AVAILABLE",
```

```
"securityPolicy": "TLS_1_2",
"policy": "...",
"routingMode" : "ROUTING_RULE_THEN_BASE_PATH_MAPPING"
}
```

Reglas de enrutamiento para conectar las etapas de API a un nombre de dominio personalizado para las API de REST

Una regla de enrutamiento es un conjunto de condiciones que, cuando coinciden, invocan una acción. Por ejemplo, una regla puede enrutar cualquier solicitud entrante a un nombre de dominio personalizado que contenga el encabezado `Hello:World` y contenga la ruta base `users` a la etapa `production` de una API de REST.

Las reglas se evalúan por orden de prioridad, y si establece el modo de enrutamiento en `ROUTING_RULE_THEN_API_MAPPING`, API Gateway siempre evalúa todas las reglas de enrutamiento antes de evaluar cualquier asignación de API. En la siguiente lista se describe cómo una regla de enrutamiento utiliza condiciones, acciones y prioridades.

Condiciones

Cuando se cumplen las condiciones de una regla, se llevan a cabo sus acciones. API Gateway admite hasta dos condiciones de encabezado y una condición de ruta. API Gateway evalúa conjuntamente las condiciones de encabezado y las condiciones de ruta base.

Puede crear una regla sin condiciones. Cuando API Gateway evalúa esta regla, siempre se realiza la acción. Puede crear una regla sin ninguna condición como regla catch-all.

Para obtener más información sobre las condiciones de encabezado, consulte the section called “Condiciones de encabezado de coincidencia”. Para obtener más información sobre las condiciones de ruta, consulte the section called “Coincidencia de condiciones de ruta base”.

Acciones

Las acciones son el resultado de la coincidencia de condiciones con una regla de enrutamiento. Actualmente, la única acción admitida es invocar una etapa de una API de REST.

Cada regla puede tener una acción.

Prioridad

La prioridad determina en qué orden se evalúan las reglas, del valor más bajo al más alto. Las reglas no pueden tener la misma prioridad.

Puede establecer la prioridad entre 1 y 1 000 000. Si una regla tiene una prioridad de uno, API Gateway la evalúa en primer lugar. Le recomendamos que, cuando cree una regla, agregue espacios en las prioridades. Esto lo ayuda a cambiar la prioridad de las reglas y a agregar nuevas reglas. Para obtener más información, consulte the section called “Cambio de la prioridad de una regla de enrutamiento”.

Para ver ejemplos de cómo API Gateway evalúa las reglas de enrutamiento, consulte the section called “Ejemplos de cómo API Gateway evalúa las reglas de enrutamiento”.

Tipos de condiciones de las reglas de enrutamiento de API Gateway

En la siguiente sección se describen los tipos de condiciones de las reglas de enrutamiento. API Gateway solo hace coincidir una regla si se cumplen todas las condiciones.

Condiciones de encabezado de coincidencia

Al crear una condición de encabezado, puede hacer coincidir el nombre del encabezado y el valor glob del encabezado, como `Hello:World`. API Gateway utiliza una coincidencia literal para validar las condiciones de encabezado de coincidencia. La condición puede utilizar hasta dos encabezados con `AND` entre ellos. Por ejemplo, la condición puede coincidir si una solicitud entrante contiene `Hello:World` y `x-version:beta`.

La coincidencia del nombre del encabezado no distingue entre mayúsculas y minúsculas, pero el valor glob del encabezado sí. `Hello:World` coincidirá con `hello:World`, pero no con `Hello:world`.

Para ver una lista de valores de encabezado restringidos, consulte the section called “Restricciones”.

Uso de caracteres comodín con condiciones de encabezado

Solo puede utilizar caracteres comodín en el valor glob del encabezado y el carácter comodín debe ser `*prefix-match`, `suffix-match*` o `*contains*`. En la siguiente tabla se muestran ejemplos de cómo utilizar comodines para la coincidencia de condiciones de encabezado.

Condiciones de encabezado	Solicitudes que coinciden con la regla de enrutamiento	Solicitudes que no coinciden con la regla de enrutamiento
`x-version: a*`	• `x-version: account` • `x-version: alpha`	• `x-version: backup` • `x-version: beta`

Condiciones de encabezado	Solicitudes que coinciden con la regla de enrutamiento	Solicitudes que no coinciden con la regla de enrutamiento
		• `x-version: users`
`x-version: *a`	• `x-version: alpha` • `x-version: beta`	• `x-version: account` • `x-version: backup` • `x-version: users`
`x-version: *a*`	• `x-version: account` • `x-version: alpha` • `x-version: backup` • `x-version: beta`	• `x-version: users`
`x-version: *a*` y `x-version: *b*`	• `x-version: backup` • `x-version: beta`	• `x-version: account` • `x-version: alpha` • `x-version: users`
`x-version: b*` y `x-version: *a`	• `x-version: beta`	• `x-version: account` • `x-version: alpha` • `x-version: backup` • `x-version: users`
`x-version: *`	• `x-version: account` • `x-version: alpha` • `x-version: backup` • `x-version: beta` • `x-version: users`	Ninguno

Si crea condiciones para varios valores de encabezado, como `Accept:application/json,text/xml`, le recomendamos que utilice `*contains*` para las condiciones de encabezado y evite crear condiciones utilizando el carácter de coma (`,`).

Dado que API Gateway enruta las condiciones de encabezado literalmente, las coincidencias semánticas podrían enrutarse de forma diferente. En la siguiente tabla se muestra la diferencia en los resultados de las reglas de enrutamiento.

Condiciones de encabezado	Solicitudes que coinciden con la regla de enrutamiento	Solicitudes que no coinciden con la regla de enrutamiento
`Accept: *json`	• `Accept:application/json Accept:text/xml`	• `Accept:application/json,text/xml`
`Accept: *json*`	• `Accept:application/json Accept:text/xml` • `Accept:application/json,text/xml`	Ninguno

Coincidencia de condiciones de ruta base

Al crear una condición de ruta base, si la solicitud entrante contiene la ruta especificada, la regla coincide. La coincidencia distingue entre mayúsculas y minúsculas, por lo que la ruta `New/Users` no coincidirá con `new/users`.

Puede crear una condición de ruta base solo para una ruta base.

Para obtener una lista de condiciones de ruta base restringidas, the section called “Restricciones”.

Eliminación de la ruta base con condiciones de ruta base

Cuando crea una condición de ruta base, puede elegir eliminar la ruta base. Cuando elimina la ruta base, API Gateway elimina la ruta base coincidente entrante cuando invoca la API de destino. Este es el mismo comportamiento que cuando utiliza una asignación de API. Si no elimina la ruta base, API Gateway reenvía toda la ruta base a la API de destino. Le recomendamos que solo elimine la ruta base cuando vuelva a crear una asignación de API.

En la siguiente tabla se muestran ejemplos de cómo API Gateway evalúa la condición de eliminación de la ruta base.

Condición	Eliminación de la ruta base	Solicitud entrante	Resultado
Si la ruta base contiene `PetStoreShopper/dogs`	True	`GET https://example.com/PetStoreShopper/dogs`	API Gateway llama al método `GET` del recurso `/`.
Si la ruta base contiene `PetStoreShopper/dogs` .	False	`GET https://example.com/PetStoreShopper/dogs`	API Gateway llama al método `GET` del recurso `PetStoreShopper/dogs` .
Si la ruta base contiene `PetStoreShopper`	True	`GET https://example.com/PetStoreShopper/dogs`	API Gateway llama al método `GET` del recurso `dogs`.
Si la ruta base contiene `PetStoreShopper`	False	`GET https://example.com/PetStoreShopper/dogs`	API Gateway llama al método `GET` del recurso `PetStoreShopper/dogs` .
Si la ruta base contiene `PetStoreShopper`	True	`GET https://example.com/PetStoreShopper?birds=available`	API Gateway llama al método `GET` del recurso `/` con el parámetro de cadena de consulta `birds=available` .
Si la ruta base contiene `PetStoreShopper`	False	`GET https://example.com/PetStoreShopper?birds=available`	API Gateway llama al método `GET` del recurso `/PetStoreShopper` con el parámetro de cadena de consulta `birds=available` .

Restricciones

- La API de destino y el nombre de dominio personalizado deben estar en la misma cuenta de AWS.
- Cada regla puede tener una API de destino.
- Solo puede crear una regla de enrutamiento para un nombre de dominio personalizado privado hacia una API privada y para un nombre de dominio personalizado público hacia una API pública. No puede mezclar recursos públicos y privados.
- Si el nombre de dominio personalizado tiene asignaciones de API tanto a las API de REST como de HTTP, no se admiten las reglas de enrutamiento.
- El número de prioridad máximo es 1 000 000.
- Restricciones de encabezado:
  - Cada condición `anyOf` solo puede contener un valor de encabezado.
  - Los únicos caracteres permitidos para los nombres de encabezado y los valores glob de encabezado están especificados por RFC 7230, que son `a-z`, `A-Z`, `0-9` y los siguientes caracteres especiales: `*?-!#$%&'.^_`|~`.
  - Puede utilizar un comodín en el valor glob del encabezado, pero el comodín debe ser `*prefix-match`, `suffix-match*` o `*contains*`. No puede utilizar `*` en medio de un valor glob de encabezado.
  - No se admiten nombres de encabezado con caracteres comodín.
  - El nombre del encabezado debe tener menos de 40 caracteres.
  - El valor glob del encabezado debe tener menos de 128 caracteres.
  - El valor glob del encabezado para una coincidencia de infijo debe tener menos de 40 caracteres.
  - Los siguientes encabezados no se admiten como condiciones:
    - `access-control-*`
    - `apigw-*`
    - `Authorization`
    - `Connection`
    - `Content-Encoding`
    - `Content-Length`
    - `Content-Location`
    - `Forwarded`
    - `Keep-Alive`

    - `Origin`
    - `Proxy-Authenticate`
    - `Proxy-Authorization`
    - `TE`
    - `Trailers`
    - `Transfer-Encoding`
    - `Upgrade`
    - `x-amz-*`
    - `x-amzn-*`
    - `x-apigw-api-id`
    - `X-Forwarded-For`
    - `X-Forwarded-Host`
    - `X-Forwarded-Proto`
    - `x-restAPI`
    - `Via`
- Restricciones de la ruta base:
    - La longitud de la ruta base debe ser inferior a 128 caracteres.
    - La ruta base debe contener solo letras, números y los siguientes caracteres: `$-_.+!*'()/`.

      Estos caracteres no se admiten en expresiones regulares (regex).
    - La ruta base no puede empezar ni terminar por el carácter de barra invertida (\).

Ejemplos de cómo API Gateway evalúa las reglas de enrutamiento

A continuación se muestran cuatro ejemplos de cómo API Gateway evalúa las reglas de enrutamiento y las asignaciones de API.

Ejemplo 1: solo reglas de enrutamiento

En este ejemplo, el nombre de dominio personalizado `https://petstore.example.com` tiene el modo de enrutamiento establecido en `ROUTING_RULE_ONLY` y tiene las siguientes reglas de enrutamiento y prioridades.

ID de regla	Prioridad	Condiciones	Action
`abc123`	10	Si la solicitud contiene el encabezado `Hello:World`	API de destino 1
`zzz000`	50	Si la solicitud contiene los encabezados `Accept:image/webp` y `Pet:Dog-*` y si la ruta base contiene `PetStoreShopper`	API de destino 2
`efg456`	100	Ninguno	API de destino 3

En la siguiente tabla se muestra cómo API Gateway aplica las reglas de enrutamiento anteriores a las solicitudes de ejemplo.

Solicitud	API seleccionada	Explicación
`https://petstore.example.com -h "Hello:World"`	API de destino 1	La solicitud coincide con la regla de enrutamiento `abc123`.
`https://petstore.example.com/PetStoreShopper -h "Hello:World", "Pet:Dog-Bella", "Accept:image/webp"`	API de destino 1	API Gateway evalúa todas las reglas de enrutamiento por orden de prioridad. La regla de enrutamiento `abc123` tiene la primera prioridad y las condiciones coinciden, por lo que API Gateway invoca la API de destino 1. Aunque las condiciones de la solicitud también coinciden con la regla de enrutamie

Solicitud	API seleccionada	Explicación
		nto `zzz000`, API Gateway no evalúa ninguna otra regla de enrutamiento después de establecer una coincidencia.
`https://petstore.example.com/PetStoreShopper -h "Pet:Dog-Bella", "Accept:image/webp"`	API de destino 2	La solicitud coincide con la regla de enrutamiento `zzz000`. Esta coincidencia se debe a que la cadena `Pet:Dog-Bella` coincidía con `Pet:Dog-*`
`https://petstore.example.com/PetStoreShopper -h "Pet:Dog-Bella"`	API de destino 3	La solicitud no coincide con la regla de enrutamiento `abc123`. La solicitud no coincide con la regla de enrutamiento `zzz000` porque no están presentes todos los encabezados requeridos. La siguiente regla de prioridad coincide con todas las solicitudes entrantes, por lo que API Gateway invoca la API de destino 3.

Ejemplo 2: reglas de enrutamiento y asignaciones de API

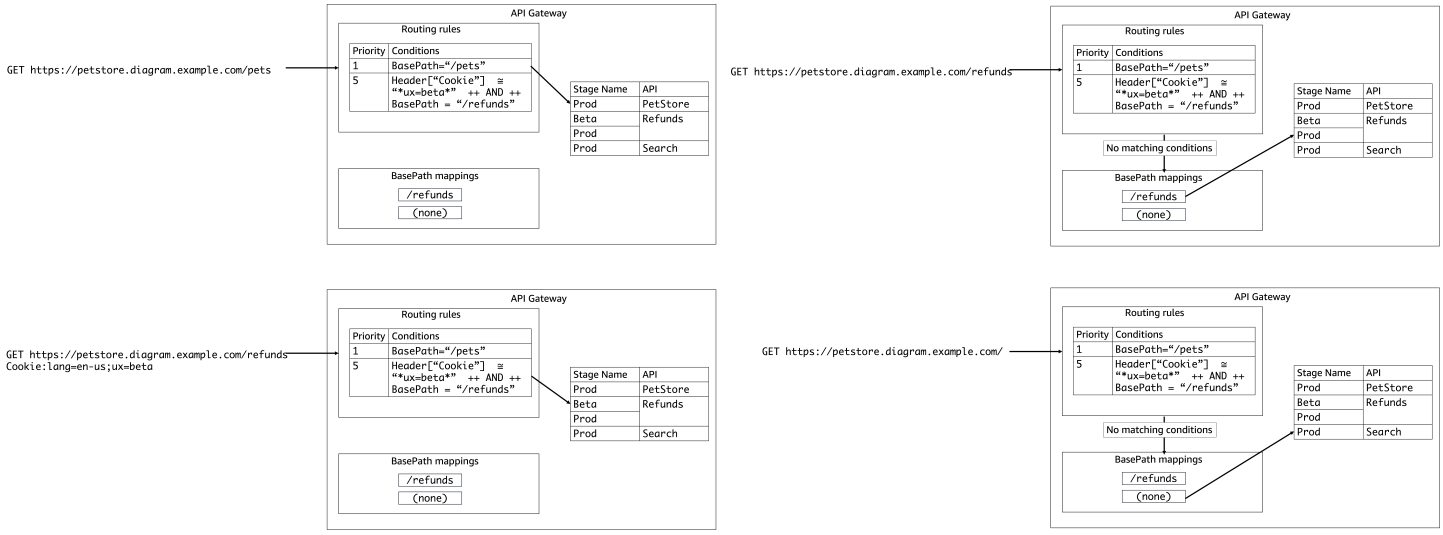
En este ejemplo, el nombre de dominio personalizado `https://petstore.diagram.example.com` tiene el modo de enrutamiento establecido en `ROUTING_RULE_THEN_API_MAPPING` y tiene las siguientes reglas de enrutamiento y asignaciones de API.

ID de regla	Prioridad	Condiciones	Action
`abc123`	1	Si la base de la solicitud contiene `pets`	Invoque la etapa `Prod` de la API `PetStore`.
`000zzz`	5	Si la solicitud contiene los encabezados `Cookie:*ux=beta*` y si la ruta base contiene `/refunds`	Invoque la etapa `Beta` de la API `Refunds`.

En la siguiente tabla se muestran las asignaciones de la API para `https://petstore.backup.example.com`.

Mapeo de API	API seleccionada
`/refunds`	Invoque la etapa `Prod` de la API `Refunds`.
`(none)`	Invoque la etapa `Prod` de la API `Search`.

En el siguiente diagrama se muestra cómo API Gateway aplica las reglas de enrutamiento anteriores a las solicitudes de ejemplo. Las solicitudes de ejemplo se resumen en la tabla que sigue a este diagrama.



En la siguiente tabal se muestra cómo API Gateway aplica las reglas de enrutamiento anteriores a las solicitudes de ejemplo.

Solicitud	API seleccionada	Explicación
`https://petstore.diagram.com/pets`	La etapa `Prod` de la API `PetStore`.	La solicitud coincide con la regla de enrutamiento `abc123`.
`https://petstore.diagram.example.com/refunds -h "Cookie:lang=en-us;ux=beta"`	La etapa `Beta` de la API `Refunds`.	La solicitud coincide con la regla de enrutamiento `000zzz`. El encabezado `Cookie` contiene la coincidencia `*contains*` correcta y la coincidencia de ruta base para esta condición.
`https://petstore.diagram.example.com/refunds`	La etapa `Prod` de la API `Refunds`.	La solicitud no tiene los encabezados necesarios para la coincidencia con la regla de enrutamiento `zzz000`. Si API Gateway no puede hacer coincidir correctamente una regla de enrutamiento, recurre a las asignaciones de API. API Gateway puede asignar la ruta base a la etapa `Prod` de la API `Refunds`.
`https://petstore.diagram.example.com/`	La etapa `Prod` de la API `Search`.	La solicitud coincide con la asignación de la API a la ruta vacía `(none)`.

Ejemplo 3: reglas de enrutamiento y asignaciones de API con varios niveles

En este ejemplo, el nombre de dominio personalizado `https://petstore.backup.example.com` tiene el modo de enrutamiento establecido en

ROUTING_RULE_THEN_API_MAPPING y tiene las siguientes reglas de enrutamiento y asignaciones de API.

En la siguiente tabla se muestran las reglas de enrutamiento para https://petstore.backup.example.com.

ID de regla	Prioridad	Condiciones	Action
abc123	10	Si la solicitud contiene el encabezado Hello:World	API de destino 1
000zzz	50	Si la solicitud contiene los encabezados Accept:image/webp y Pet:Dog-* y si la ruta base contiene PetStoreShopper	API de destino 2

En la siguiente tabla se muestran las asignaciones de la API para https://petstore.backup.example.com.

Mapeo de API	API seleccionada
PetStoreShopper	API de destino 3
PetStoreShopper/cats	API de destino 4

En la siguiente tabal se muestra cómo API Gateway aplica las reglas de enrutamiento anteriores a las solicitudes de ejemplo.

Solicitud	API seleccionada	Explicación
https://petstore.example.com/PetStor	API de destino 3	La solicitud no tiene los encabezados necesarios para la coincidencia con la regla

Solicitud	API seleccionada	Explicación
`eShopper -h "Accept:image/webp", "Pet:Cats"`		de enrutamiento `zzz000`. Si API Gateway no puede hacer coincidir correctamente una regla de enrutamiento, recurre a las asignaciones de API. API Gateway puede asignar la ruta base a la API de destino 3.
`https://petstore.example.com/PetStoreShopper/cats -h "Hello:World"`	API de destino 1	La solicitud coincide con la regla de enrutamiento `abc123`. Si el modo de enrutamiento se establece en `ROUTING_RULE_THEN_API_MAPPING` , las reglas de enrutamiento siempre tienen prioridad sobre las asignaciones de API.
`https://petstore.example.com/Admin -h "Pet:Dog-Bella"`	Ninguno	La solicitud no coincide con ninguna regla de enrutamiento ni asignación de API. Dado que no hay ninguna regla de enrutamiento predeterminada, API Gateway rechaza la llamada y envía al intermediario un código de estado `403 Forbidden` .

Ejemplo 4: reglas de enrutamiento para nombres de dominio comodín

En este ejemplo, el nombre de dominio personalizado `https://*.example.com` es un nombre de dominio comodín. El comodín admite todos los subdominios que enrutan al mismo dominio. Las siguientes reglas de enrutamiento de ejemplo cambian este comportamiento para permitir que los subdominios enruten a diferentes API de destino mediante el encabezado `Host`.

En la siguiente tabla se muestran las reglas de enrutamiento para `https://*.example.com`.

ID de regla	Prioridad	Condiciones	Action
`abc123`	10	Si la solicitud contiene el encabezado `Host:a.example.com`	API de destino 1
`000zzz`	50	Si la solicitud contiene encabezados `Host:b.example.com`	API de destino 2
`efg456`	500	Ninguno	API de destino 3

En la siguiente tabla se muestra cómo API Gateway aplica las reglas de enrutamiento anteriores a las solicitudes de ejemplo.

Solicitud	API seleccionada	Explicación
`https://a.example.com`	API de destino 1	El encabezado `Host` es `a.example.com` . Esta solicitud coincide con la regla de enrutamiento `abc123`.
`https://b.example.com`	API de destino 2	El encabezado `Host` es `b.example.com` . Esta solicitud coincide con la regla de enrutamiento `000zzz`.
`https://testing.example.com`	API de destino 3	Coincide con la regla de enrutamiento catch-all `efg456`.

Cómo utilizar las reglas de enrutamiento

Puede crear una regla de enrutamiento mediante la Consola de administración de AWS, la AWS CLI o cualquier AWS SDK. Después de crear una regla, puede cambiar la prioridad.

Creación de una regla de enrutamiento

El siguiente procedimiento muestra cómo crear una regla de enrutamiento para un nombre de dominio personalizado con un modo de enrutamiento establecido en `ROUTING_RULE_THEN_API_MAPPING` o `ROUTING_RULE_ONLY`.

Consola de administración de AWS

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija Custom domain names (Nombres de dominios personalizados) en el panel de navegación principal.
3. Elija un nombre de dominio personalizado.
4. En la pestaña Detalles de enrutamiento, elija Agregar regla de enrutamiento.
5. Elija Agregar una nueva condición para agregar una nueva condición.

   Puede agregar una condición de encabezado o de ruta base. Para hacer coincidir todas las solicitudes entrantes con el nombre de dominio personalizado, no agregue una condición.
6. En Acción, utilice la lista desplegable para seleccionar la API de destino y la etapa de destino.
7. Elija Siguiente.
8. En el campo de prioridad, introduzca un número para la prioridad.

   API Gateway evalúa las reglas en orden de prioridad, del valor más bajo al más alto.

   Si va a crear una regla sin una condición, le recomendamos que utilice un valor de prioridad alto.
9. Elija Crear regla de enrutamiento.

AWS CLI

El siguiente comando create-routing-rule crea una regla de enrutamiento con una prioridad de 50. En este ejemplo, API Gateway enruta cualquier solicitud entrante que tenga los encabezados `Hello:World` y `x-version:beta` y la ruta base `PetStoreShopper` a la API de destino `a1b2c3`.

```
aws apigatewayv2 create-routing-rule \
 --domain-name 'api.example.com' \
 --priority 50 \
```

```
  --conditions '[
    {
      "MatchHeaders": {
        "AnyOf": [
          {
            "Header": "Hello",
            "ValueGlob": "World"
          }
        ]
      }
    },
    {
      "MatchHeaders": {
        "AnyOf": [
          {
            "Header": "x-version",
            "ValueGlob": "beta"
          }
        ]
      }
    },
    {
      "MatchBasePaths": {
        "AnyOf": [
          "PetStoreShopper"
        ]
      }
    }
  ]'\
  --actions '[
  {
    "InvokeApi": {
      "ApiId": "a1b2c3",
      "Stage": "prod"
    }
  }
 ]'
```

El resultado será similar al siguiente.

```
{
    "Actions": [
        {
```

```
            "InvokeApi": {
                "ApiId": "a1b2c3",
                "Stage": "prod",
                "StripBasePath": false
            }
        }
    ],
    "Conditions": [
        {
            "MatchHeaders": {
                "AnyOf": [
                    {
                        "Header": "Hello",
                        "ValueGlob": "World"
                    }
                ]
            }
        },
        {
            "MatchHeaders": {
                "AnyOf": [
                    {
                        "Header": "x-version",
                        "ValueGlob": "beta"
                    }
                ]
            }
        },
        {
            "MatchBasePaths": {
                "AnyOf": [
                    "PetStoreShopper"
                ]
            }
        }
    ],
    "Priority": 50,
    "RoutingRuleArn": "arn:aws:apigateway:us-west-2:111122223333:/domainnames/
api.example.com/routingrules/abc123",
    "RoutingRuleId": "abc123"
}
```

Cambio de la prioridad de una regla de enrutamiento

Puede cambiar la prioridad de una regla de enrutamiento. Esto tiene efecto inmediato y podría afectar el modo en que los consumidores de API invocan los nombres de dominio personalizados. Le recomendamos que, cuando establezca las prioridades de las reglas de enrutamiento, deje espacios entre las reglas.

Por ejemplo, considere dos reglas de enrutamiento, la regla `abc123` con una prioridad de 50 y la regla `zzz000` con una prioridad de 150. Para cambiar la prioridad de las reglas de modo que API Gateway evalúe primero la regla `zzz000`, puede cambiar la prioridad de la regla `zzz000` a 30.

Consola de administración de AWS

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija Custom domain names (Nombres de dominios personalizados) en el panel de navegación principal.
3. Elija un nombre de dominio personalizado.
4. En la pestaña Detalles de enrutamiento, elija la regla de enrutamiento y, a continuación, Editar.
5. Elija Siguiente.
6. Para la prioridad, introduzca la nueva prioridad.
7. Seleccione Save changes (Guardar cambios).

AWS CLI

El siguiente comando put-routing-rule cambia la prioridad de una regla de enrutamiento `abc123`.

```
aws apigatewayv2 put-routing-rule \
 --domain-name 'api.example.com' \
 --priority 30 \
 --routing-rule-id abc123 \
 --conditions '[
   {
     "MatchHeaders": {
       "AnyOf": [
         {
           "Header": "Hello",
           "ValueGlob": "World"
         }
```

```
        ]
      }
    },
    {
      "MatchHeaders": {
        "AnyOf": [
          {
            "Header": "x-version",
            "ValueGlob": "beta"
          }
        ]
      }
    },
    {
      "MatchBasePaths": {
        "AnyOf": [
          "PetStoreShopper"
        ]
      }
    }
  ]'\
  --actions '[
  {
    "InvokeApi": {
      "ApiId": "a1b2c3",
      "Stage": "prod"
    }
  }
 ]'
```

El resultado será similar al siguiente:

```
{
    "Actions": [
        {
            "InvokeApi": {
                "ApiId": "a1b2c3",
                "Stage": "prod",
                "StripBasePath": false
            }
        }
    ],
    "Conditions": [
```

```
        {
            "MatchHeaders": {
                "AnyOf": [
                    {
                        "Header": "Hello",
                        "ValueGlob": "World"
                    }
                ]
            }
        },
        {
            "MatchHeaders": {
                "AnyOf": [
                    {
                        "Header": "x-version",
                        "ValueGlob": "beta"
                    }
                ]
            }
        },
        {
            "MatchBasePaths": {
                "AnyOf": [
                    "PetStoreShopper"
                ]
            }
        }
    ],
    "Priority": 38,
    "RoutingRuleArn": "arn:aws:apigateway:us-west-2:111122223333:/domainnames/
api.example.com/routingrules/abc123",
    "RoutingRuleId": "abc123"
}
```

Nueva creación de una asignación de API mediante reglas de enrutamiento

Puede volver a crear una asignación de API mediante reglas de enrutamiento. Para volver a crear una asignación de API, asegúrese de activar la separación de rutas base. Esto preserva el comportamiento de las asignaciones de API. Para obtener más información, consulte the section called “Eliminación de la ruta base con condiciones de ruta base”.

En el siguiente tutorial se muestra cómo volver a crear la asignación de API `https://api.example.com/orders/v2/items/categories/5` como una regla de enrutamiento y cómo actualizar los registros de acceso para registrar el ID de la regla de enrutamiento que API Gateway utiliza para enviar tráfico a la API.

Consola de administración de AWS

Establecimiento del modo de enrutamiento en ROUTING_RULE_THEN_API_MAPPING

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija Custom domain names (Nombres de dominios personalizados) en el panel de navegación principal.
3. Elija el nombre de dominio personalizado.
4. En Detalles del dominio, elija Editar.
5. En Modo de enrutamiento, elija ROUTING_RULE_THEN_API_MAPPING.
6. Seleccione Save.

Una vez establecido el modo de enrutamiento, cree la regla de enrutamiento.

Creación de la regla de enrutamiento

1. En la pestaña Detalles de enrutamiento, elija Agregar regla de enrutamiento.
2. Seleccione Agregar nueva condición y, a continuación, elija Ruta.
3. En Ruta, introduzca **`orders/v2/items/categories/5`**.
4. En Eliminar ruta base, elija Activa.
5. En API de destino, elija la API de destino.
6. En Etapa de destino, elija la etapa de destino.
7. Elija Siguiente.
8. Para la prioridad, introduzca una prioridad.

   Aunque mantenga la asignación de API existente, API Gateway utilizará siempre la nueva regla de enrutamiento, ya que las reglas de enrutamiento siempre tienen prioridad sobre las asignaciones de API.
9. Seleccione Save changes (Guardar cambios).

Después de crear la regla de enrutamiento, actualice el formato del registro de acceso de la etapa o cree un nuevo registro para confirmar que API Gateway utiliza la regla de enrutamiento para enviar tráfico a la API.

Actualización de registros de acceso

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija la API.
3. En el panel de navegación principal, elija Etapas.
4. En Registros y seguimiento, elija Editar.

   Si no dispone de un grupo de registro, consulte the section called “CloudWatch Logs”.
5. Agregue **`$context.customDomain.routingRuleIdMatched`** al formato de registro.

   Este grupo de registro guarda el ID de la regla de enrutamiento que API Gateway utilizó para enviar tráfico a la API. Para obtener más información, consulte the section called “No puedo saber cómo API Gateway ha enviado tráfico a mis API”.
6. Seleccione Save.

Después de actualizar los registros de acceso, invoque el nombre de dominio personalizado. El siguiente es un comando curl de ejemplo para invocar el nombre de dominio personalizado `https://api.example.com` con la ruta base `orders/v2/items/categories/5`.

```
curl "https://api.example.com/orders/v2/items/categories/5"
```

Una vez que haya invocado correctamente el nombre de dominio personalizado, confirme que CloudWatch Logs muestra `routingRuleIdMatched`. Para obtener información sobre cómo utilizar la consola de CloudWatch Logs para ver un grupo de registros, consulte the section called “Visualización de eventos de registro en la consola de CloudWatch”.

AWS CLI

1. Utilice el siguiente comando update-domain-name para actualizar el nombre de dominio *`api.example.com`* y utilizar el modo de enrutamiento `ROUTING_RULE_THEN_API_MAPPING`.

   ```
   aws apigatewayv2 update-domain-name \
     --domain-name 'api.example.com' \
   ```

```
  --routing-mode ROUTING_RULE_THEN_API_MAPPING
```

2. Utilice el siguiente comando create-routing-rule para crear una nueva regla de enrutamiento para volver a crear la asignación de la API `https://api.example.com/orders/v2/items/categories/5`.

```
aws apigatewayv2 create-routing-rule \
  --domain-name 'api.example.com' \
  --priority 50 \
  --conditions '[
  {
    "MatchBasePaths": {
      "AnyOf": [
        "orders/v2/items/categories/5"
      ]
    }
  }
]' \
  --actions '[
  {
    "InvokeApi": {
      "ApiId": "a1b2c3",
      "Stage": "prod",
      "StripBasePath": true
    }
  }
]'
```

3. Utilice el siguiente comando update-stage para actualizar el formato de los registros de acceso e incluir la variable `$context.customDomain.routingRuleIdMatched`. Esta variable registra el ID de la regla de enrutamiento que API Gateway utilizó para enviar tráfico a la API. Utilice este registro para confirmar que API Gateway utiliza la regla de enrutamiento para enviar tráfico a la API. Para obtener más información, consulte the section called “No puedo saber cómo API Gateway ha enviado tráfico a mis API”.

```
aws apigateway update-stage \
  --rest-api-id a1bc2c3 \
  --stage-name prod \
  --patch-operations "op=replace,path=/accessLogSettings/format,value='\
$context.path \$context.customDomain.routingRuleIdMatched \$context.requestId \
$context.extendedRequestId'"
```

Si no dispone de un grupo de registro, consulte the section called “CloudWatch Logs”.

4. Utilice el siguiente comando curl de ejemplo para invocar el nombre de dominio personalizado con la ruta base `orders/v2/items/categories/5`.

```
curl "https://api.example.com/orders/v2/items/categories/5
```

5. Utilice el siguiente comando filter-log-events para obtener los eventos de registro del grupo de registro `access-log-group-orders` que contengan el ID de regla de enrutamiento `abc123`.

```
aws logs filter-log-events --log-group-name access-log-group-orders --filter-
pattern abc123
```

Esto confirma que API Gateway utilizó la regla de enrutamiento para enviar tráfico a la API.

Solución de problemas con las reglas de enrutamiento

La siguiente guía para la solución de problemas puede ayudarlo a resolver problemas con las reglas de enrutamiento.

No puedo saber cómo API Gateway ha enviado tráfico a mis API

Puede utilizar los registros de acceso de la etapa de API de REST para registrar y solucionar problemas de las reglas de enrutamiento. Puede ver el ID de la regla de enrutamiento que API Gateway utilizó para enviar tráfico a la API mediante la variable `$context.customDomain.routingRuleIdMatched`. Para ver la asignación de API que API Gateway utilizó para enviar tráfico a la API, utilice la variable `$context.customDomain.basePathMatched`.

Para registrar las reglas de enrutamiento, debe configurar un ARN de rol de CloudWatch Logs apropiado para la cuenta y crear un grupo de registro.

El siguiente ejemplo de grupo de registro de acceso puede recuperar la información pertinente para la solución de problemas de reglas de enrutamiento y asignaciones de API. API Gateway solo rellena la variable de contexto para el mecanismo de enrutamiento que utilizó; de lo contrario, la variable de contexto es `-`.

CLF

```
$context.path $context.customDomain.routingRuleIdMatched
 $context.customDomain.basePathMatched $context.requestId $context.extendedRequestId
```

JSON

```
{"requestPath": "$context.path", "routingRuleId" :
 "$context.customDomain.routingRuleIdMatched", "API mapping" :
 "$context.customDomain.basePathMatched", "requestId":"$context.requestId",
 "extendedRequestId":"$context.extendedRequestId"}
```

XML

```
<request id="$context.requestId"> <requestPath>$context.path</requestPath>
 <ruleId>$context.customDomain.routingRuleIdMatched</ruleId> <ApiMapping>
$context.customDomain.basePathMatched</ApiMapping> <extendedRequestId>
$context.extendedRequestId</extendedRequestId> </request>
```

CSV

```
$context.path,$context.customDomain.routingRuleIdMatched,
$context.customDomain.basePathMatched,$context.requestId,$context.extendedRequestId
```

También le recomendamos que confirme el modo de enrutamiento para el nombre de dominio personalizado. Para obtener más información, consulte the section called “Establecimiento del modo de enrutamiento para el nombre de dominio personalizado”.

No puedo habilitar las reglas de enrutamiento en mi nombre de dominio personalizado

Es posible que reciba el siguiente error de API Gateway:

```
Your account doesn’t have permission to use RoutingRules.
This might be caused by an IAM policy in your account with a deny statement on
 BasePathMapping or ApiMapping.
To grant permission for this account to use RoutingRules, use the UpdateAccount API.
This will impact any existing IAM policies that deny access to BasePathMapping or
 ApiMapping.
See API Gateway documentation for further details.
```

Recibirá este error si tiene o tenía una política de IAM que deniega el acceso a BasePathMapping o ApiMapping. Cuando habilite reglas de enrutamiento para un nombre de dominio personalizado, aunque la política seguirá denegando el acceso a `BasePathMapping` o `ApiMapping`, la misma política puede utilizarse para acceder a `RoutingRule`. Esto podría permitir a un usuario cambiar el comportamiento de enrutamiento del nombre de dominio personalizado.

Por ejemplo, si tuviera una política como la siguiente:

```
{
    "Sid": "DenyCreatingApiMappings",
    "Effect": "Deny",
    "Action": "apigateway:POST",
    "Resource": [
        "arn:aws:apigateway:us-west-2::/domainnames/example.com/apimappings"
    ]
}
```

Cuando habilite las reglas de enrutamiento para `example.com`, esta política seguirá denegando el acceso a la creación de `ApiMapping` pero no denegará el acceso a la creación de `RoutingRule`.

Le recomendamos que audite las políticas de IAM de la cuenta. La siguiente política de ejemplo denegará el acceso a la creación de `ApiMapping`, `BasePathMapping` y `RoutingRule`:

```
{
    "Sid": "DenyCreatingBasePathMappingsApiMappings",
    "Effect": "Deny",
    "Action": "apigateway:POST",
    "Resource": [
        "arn:aws:apigateway:us-west-2::/domainnames/example.com/basepathmappings",
        "arn:aws:apigateway:us-west-2::/domainnames/example.com/apimappings"
    ]
},
{
    "Sid": "DenyCreatingRoutingRules",
    "Effect": "Deny",
    "Action": "apigateway:CreateRoutingRule",
    "Resource": [
        "arn:aws:apigateway:us-west-2:111122223333:/domainnames/example.com/
routingrules/*"
    ]
}
```

Una vez que haya confirmado que todas las políticas se han actualizado, puede actualizar la configuración de cuenta de la API para habilitar las reglas de enrutamiento para una región.

Utilice el siguiente comando update-account para actualizar la configuración de cuenta de la API para una región:

```
aws apigateway update-account --patch-operations 'op=remove,path=/
features,value=BlockedForRoutingRules' --region us-west-2
```

Después de actualizar la configuración de cuenta de la API, puede cambiar el modo de enrutamiento del nombre de dominio personalizado. También puede seguir utilizando las políticas de IAM para denegar el acceso a `RoutingRules`, `ApiMapping` o `BasePathMapping`.

Uso de las asignaciones de API para conectar etapas de API a un nombre de dominio personalizado para las API de REST

Las asignaciones de API se utilizan para conectar etapas de la API a un nombre de dominio personalizado. Esto envía el tráfico a sus API a través de su nombre de dominio personalizado.

Una asignación de API especifica una API, una etapa y, de forma opcional, una ruta que se utilizará para la asignación. Por ejemplo, puede asignar `https://api.example.com/orders` a la etapa `production` de una API.

Puede asignar etapas de API HTTP y de API de REST al mismo nombre de dominio personalizado.

Antes de crear una asignación de API, debe tener una API, una etapa y un nombre de dominio personalizado. Para obtener más información sobre cómo crear un nombre de dominio personalizado, consulte the section called “Configuración de un nombre de dominio personalizado regional”.

Solicitudes entrantes para el nombre de dominio personalizado

Cuando asigna un nombre de dominio personalizado a una etapa de la API, API Gateway elimina la ruta base entrante. Esto elimina la ruta base asignada de la invocación a la API. Por ejemplo, si la asignación de ruta base fuera `https://api.example.com/orders/shop/5` para la etapa `test` y utilizara la siguiente solicitud, `https://api.example.com/orders/shop/5/hats`, API Gateway invocaría el recurso `/hats` de la etapa `test` de la API, no el recurso `orders/shop/5/hats`.

Asignación de solicitudes a la API

A continuación se explica cómo API Gateway evalúa las asignaciones de la API.

Puede crear una asignación de la API mediante asignaciones de un solo nivel, como una asignación de la API desde `orders` a la etapa `beta` de una API y una asignación de la API desde `shipping` a la etapa `alpha` de una API. En el caso de los nombres de dominio personalizados regionales con la política de seguridad TLS 1.2, API Gateway admite asignaciones de la API de varios niveles. Puede crear una asignación de la API de `orders/v1/items` a la etapa `alpha` de una API y de `orders/v2/items` a la etapa `beta` de una API. Cuando crea una asignación con varios niveles, API Gateway envía las solicitudes a la asignación de la API que tenga la ruta de coincidencia más larga.

Puede crear una asignación de la API a la ruta vacía (`(none)`). Si ninguna ruta coincide con la solicitud, API Gateway envía la solicitud a la ruta vacía (`(none)`).

En este ejemplo, el nombre de dominio personalizado `https://api.example.com` tiene las siguientes asignaciones de la API.

Asignación de API	API seleccionada
`(none)`	API 1
`orders`	API 2
`orders/v1/items`	API 3
`orders/v2/items`	API 4
`orders/v1/items/categories`	API 5

En la siguiente tabla se muestra cómo API Gateway aplica las asignaciones de API anteriores a las solicitudes de ejemplo.

Solicitud	API seleccionada	Explicación
`https://api.example.com/orders`	API 2	La solicitud coincide exactamente con esta asignación de API.

Solicitud	API seleccionada	Explicación
`https://api.example.com/orders/v1/items`	API 3	La solicitud coincide exactamente con esta asignación de API.
`https://api.example.com/orders/v2/items`	API 4	La solicitud coincide exactamente con esta asignación de API.
`https://api.example.com/orders/v1/items/123`	API 3	API Gateway elige la asignación con la ruta de coincidencia más larga. El `123` al final de la solicitud no afecta a la selección. Consulte the section called “Solicitudes entrantes para el nombre de dominio personalizado”.
`https://api.example.com/orders/v1/items/categories/5`	API 5	API Gateway elige la asignación con la ruta de coincidencia más larga.
`https://api.example.com/customers`	API 1	API Gateway utiliza la asignación vacía como un catch-all.

Solicitud	API seleccionada	Explicación
`https://api.example.com/ordersandmore`	API 2	API Gateway elige la asignación con el prefijo de coincidencia más largo. En el caso de un nombre de dominio personalizado configurado con asignaciones de un solo nivel, por ejemplo, solo `https://api.example.com/orders` y `https://api.example.com/`, API Gateway elegiría `API 1`, ya que no hay una ruta que coincida con `ordersandmore`.

Restricciones

- En el caso de una asignación de API, el nombre de dominio personalizado y las API asignadas deben pertenecer a la misma cuenta de AWS.
- Las asignaciones de API deben contener solo letras, números y los siguientes caracteres: `$-_.+!*'()/`.
- La longitud máxima de la ruta en una asignación de API es de 300 caracteres.
- Puede tener 200 asignaciones de API con varios niveles para cada nombre de dominio. Este límite no incluye la asignación de la API con niveles únicos, como `/prod`.
- Solo puede asignar API HTTP a un nombre de dominio regional personalizado con la política de seguridad de TLS 1.2.
- No puede asignar las API de WebSocket al mismo nombre de dominio personalizado que una API HTTP o API REST.
- Después de crear sus asignaciones de API, debe crear o actualizar el registro de recursos del proveedor de DNS para asignarlo al punto de conexión de la API.

- Si crea una asignación de API con varios niveles, API Gateway convierte todos los nombres de encabezado a minúsculas.

Creación de una asignación de API

Para crear una asignación de API, primero debe crear un nombre de dominio personalizado, una API y una etapa. El nombre de dominio personalizado debe tener un modo de enrutamiento establecido en `ROUTING_RULE_THEN_API_MAPPING` o `API_MAPPING_ONLY`. Para obtener información sobre cómo establecer el modo de enrutamiento, consulte the section called “Establecimiento del modo de enrutamiento para el nombre de dominio personalizado”.

Para ver ejemplos de las plantillas de AWS Serverless Application Model que crean todos los recursos, consulte Sessions With SAM en GitHub.

Consola de administración de AWS

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija Custom domain names (Nombres de dominios personalizados) en el panel de navegación principal.
3. Elija un nombre de dominio personalizado.
4. En la pestaña Detalles de enrutamiento, elija Configurar los mapeos de la API.
5. Especifique los valores de API, Etapa y Ruta para el mapeo.
6. Seleccione Save.

AWS CLI

El siguiente comando create-api-mapping permite crear una asignación de API. En este ejemplo, API Gateway envía solicitudes a `api.example.com/v1/orders` a la API y a la etapa especificadas.

> **Note**
>
> Para crear mapeos de la API con varios niveles, debe utilizar `apigatewayv2`.

```
aws apigatewayv2 create-api-mapping \
    --domain-name api.example.com \
```

```
    --api-mapping-key v1/orders \
    --api-id a1b2c3d4 \
    --stage test
```

CloudFormation

El siguiente ejemplo de CloudFormation crea un mapeo de la API.

> **Note**
>
> Para crear mapeos de la API con varios niveles, debe utilizar `AWS::ApiGatewayV2`.

```
MyApiMapping:
  Type: 'AWS::ApiGatewayV2::ApiMapping'
  Properties:
    DomainName: api.example.com
    ApiMappingKey: 'orders/v2/items'
    ApiId: !Ref MyApi
    Stage: !Ref MyStage
```

Tipos de direcciones IP para nombres de dominio personalizados para API Gateway

Al crear un nombre de dominio personalizado, se especifica el tipo de direcciones IP que puede invocar el dominio. Puede elegir IPv4 para resolver las direcciones IPv4 para invocar el dominio o puede elegir pila doble para permitir que las direcciones IPv4 e IPv6 invoquen el dominio. Le recomendamos que establezca el tipo de dirección IP en pila doble para mitigar el agotamiento del espacio IP o para mejorar la postura de seguridad. Para obtener más información sobre los beneficios de un tipo de dirección IP de pila doble, consulte IPv6 en AWS.

Puede cambiar el tipo de dirección IP actualizando la configuración de punto de conexión del nombre de dominio.

Consideraciones para los tipos de direcciones IP

Es posible que las siguientes consideraciones afecten al uso de los tipos de direcciones IP.

- El tipo de dirección IP predeterminado para los nombres de dominio personalizados de API Gateway para API públicas es IPv4.

- Los nombres de dominio personalizados privados solo pueden tener un tipo de dirección IP de pila doble.
- El nombre de dominio personalizado no necesita tener el mismo tipo de dirección IP para todas las API asignadas a él. Si desactiva el punto de conexión de la API predeterminado, esto podría afectar a la forma en que los intermediarios pueden invocar al dominio.

Cambio del tipo de dirección IP de un nombre de dominio personalizado

Puede cambiar el tipo de dirección IP actualizando la configuración de punto de conexión del nombre de dominio. Puede actualizar la configuración de punto de conexión mediante el uso de la Consola de administración de AWS, la AWS CLI, CloudFormation, o un AWS SDK.

Consola de administración de AWS

Cambio del tipo de dirección IP de un nombre de dominio personalizado

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Elija un nombre de dominio personalizado público.
3. Elija la Configuración de punto de conexión.
4. Para tipo de dirección IP, seleccione IPv4 o Pila doble.
5. Seleccione Save.

AWS CLI

El siguiente comando update-domain-name actualiza una API para que tenga un tipo de dirección IP de pila doble:

```
aws apigateway update-domain-name \
    --domain-name dualstack.example.com \
    --patch-operations "op='replace',path='/endpointConfiguration/
ipAddressType',value='dualstack'"
```

El resultado será similar al siguiente:

```
{
    "domainName": "dualstack.example.com",
    "certificateUploadDate": "2025-02-04T14:46:10-08:00",
```

```
    "regionalDomainName": "d-abcd1234.execute-api.us-east-1.amazonaws.com",
    "regionalHostedZoneId": "Z3LQWSYCGH4ADY",
    "regionalCertificateArn": "arn:aws:acm:us-east-1:111122223333:certificate/
a1b2c3d4-5678-90ab-cdef",
    "endpointConfiguration": {
        "types": [
            "REGIONAL"
        ],
        "ipAddressType": "dualstack"
    },
    "domainNameStatus": "AVAILABLE",
    "securityPolicy": "TLS_1_2",
    "tags": {}
}
```

Elección de una política de seguridad para el dominio personalizado en API Gateway

Una política de seguridad es una combinación predefinida de versión mínima de TLS y conjuntos de cifrado que ofrece API Gateway. Cuando los clientes realizan un establecimiento de comunicación de TLS en la API o el nombre de dominio personalizado, la política de seguridad aplica la versión de TLS y el conjunto de cifrado aceptado por API Gateway. Las políticas de seguridad protegen las API y los nombres de dominio personalizados de los problemas de seguridad de red como, por ejemplo, la manipulación y el espionaje entre un cliente y el servidor.

API Gateway admite políticas de seguridad heredadas y políticas de seguridad mejoradas. `TLS_1_0` y `TLS_1_2` son políticas de seguridad heredadas. Utilice estas políticas de seguridad para cargas de trabajo generalizadas o para empezar a crear una API. Cualquier política que comience con `SecurityPolicy_` es una política de seguridad mejorada. Utilice estas políticas para cargas de trabajo reguladas, una gobernanza avanzada o para utilizar la criptografía poscuántica. Al utilizar una política de seguridad mejorada, también debe configurar el modo de acceso de punto de conexión para una mayor gobernanza. Para obtener más información, consulte the section called “Modo de acceso de punto de conexión”.

Consideraciones

A continuación, se muestran consideraciones sobre las políticas de seguridad de los nombres de dominio personalizados para las API de REST en API Gateway:

- No puede habilitar el TLS mutuo en un nombre de dominio que utilice una política de seguridad mejorada.

- No puede asignar una API HTTP a un nombre de dominio que utilice una política de seguridad mejorada.
- Si habilita la asignación de rutas base de varios niveles a una API de REST que utilice una política de seguridad mejorada, no podrá crear una asignación de ruta base a una API HTTP para el mismo nombre de dominio.
- La API se puede asignar a un nombre de dominio personalizado con una política de seguridad diferente a la de la API. Al invocar ese nombre de dominio personalizado, API Gateway utiliza la política de seguridad del dominio personalizado para negociar el establecimiento de comunicación de TLS. Si desactiva el punto de conexión de la API predeterminado, esto podría afectar a la forma en que los intermediarios pueden invocar a la API.
- API Gateway admite políticas de seguridad en todas las API. Sin embargo, solo puede elegir una política de seguridad para las API de REST. API Gateway solo admite la política de seguridad `TLS_1_2` para las API HTTP o de WebSocket.
- API Gateway no admite la actualización de una política de seguridad para un nombre de dominio con varios tipos de punto de conexión. Si tiene varios tipos de punto de conexión para un nombre de dominio, elimine uno de ellos para actualizar la política de seguridad.

Cómo aplica API Gateway las políticas de seguridad

El siguiente ejemplo muestra cómo API Gateway aplica las políticas de seguridad mediante la política de seguridad `SecurityPolicy_TLS13_1_3_2025_09` como ejemplo.

La política de seguridad `SecurityPolicy_TLS13_1_3_2025_09` acepta el tráfico de TLS 1.3 y rechaza el tráfico de TLS 1.2 y TLS 1.0. Para el tráfico de TLS 1.3, la política de seguridad acepta los siguientes conjuntos de cifrado:

- `TLS_AES_128_GCM_SHA256`
- `TLS_AES_256_GCM_SHA384`
- `TLS_CHACHA20_POLY1305_SHA256`

API Gateway no acepta ningún otro conjunto de sistemas de cifrado. Por ejemplo, la política de seguridad rechazaría cualquier tráfico de TLS 1.3 que utilice el conjunto de cifrado de `AES128-SHA`.

Para supervisar qué protocolo TLS y qué cifrado utilizan los clientes para acceder a la API Gateway, puede utilizar las variables de contexto `$context.tlsVersion` y `$context.cipherSuite` en los registros de acceso. Para obtener más información, consulte the section called “Supervisión”.

Para ver las políticas de seguridad predeterminadas de todas las API de REST y los nombres de dominio personalizados, consulte the section called “Políticas de seguridad predeterminadas”.
Para ver las políticas de seguridad aceptadas de todas las API de REST y los nombres de dominio personalizados, consulte the section called “Políticas de seguridad admitidas”.

Cambio de la política de seguridad del nombre de dominio personalizado

Si cambia la política de seguridad, esta tardará unos 15 minutos en completarse. Puede supervisar el `lastUpdateStatus` del nombre de dominio personalizado. A medida que el nombre de dominio personalizado se actualice, el `lastUpdateStatus` será `PENDING` y cuando se complete será `AVAILABLE`.

Cuando utilice una política de seguridad que comience por `SecurityPolicy_`, también debe activar el modo de acceso de punto de conexión. Para obtener más información, consulte the section called “Modo de acceso de punto de conexión”.

Consola de administración de AWS

Cambio de la política de seguridad de un nombre de dominio personalizado

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Elija un nombre de dominio personalizado que envíe el tráfico a las API de REST.

   Asegúrese de que solo haya un tipo de punto de conexión asociado al nombre de dominio personalizado.
3. Elija Configuración de nombre de dominio personalizado y, a continuación, elija Editar.
4. En Política de seguridad, seleccione una nueva política.
5. Para Modo de acceso de punto de conexión, elija Estricto.
6. Seleccione Save changes (Guardar cambios).

AWS CLI

El siguiente comando update-domain-name actualiza un nombre de dominio para utilizar la política de seguridad `SecurityPolicy_TLS13_1_3_2025_09`:

```
aws apigateway update-domain-name \
    --domain-name example.com \
    --patch-operations '[
```

```
        {
            "op": "replace",
            "path": "/securityPolicy",
            "value": "SecurityPolicy_TLS13_1_3_2025_09"
        },
        {
            "op": "replace",
            "path": "/endpointAccessMode",
            "value": "STRICT"
        }
    ]'
```

El resultado será similar al siguiente:

```
{
   "domainName": "example.com",
   "endpointConfiguration": {
      "types": [ "REGIONAL" ],
      "ipAddressType": "dualstack"
   },
   "regionalCertificateArn": "arn:aws:acm:us-west-2:111122223333:certificate/
a1b2c3d4-5678-90ab-cdef",
   "securityPolicy": "SecurityPolicy_TLS13_1_3_2025_09",
   "endpointAccessMode": "STRICT"
}
```

Información acerca de las API de HTTP y las API de WebSocket

Para obtener más información sobre las API de HTTP y las API de WebSocket, consulte the section called “Política de seguridad de las API de HTTP en API Gateway” y the section called “Política de seguridad de las API de WebSocket en API Gateway”.

Deshabilitación del punto de conexión predeterminado para las API de REST

De forma predeterminada, los clientes pueden invocar su API mediante el punto de conexión `execute-api` que API Gateway genera para ella. Para asegurarse de que los clientes solo puedan acceder a su API mediante un nombre de dominio personalizado, deshabilite el punto de conexión predeterminado `execute-api`. Los clientes pueden seguir conectándose al punto de conexión predeterminado, pero recibirán un código de estado `403 Forbidden`. La desactivación del punto de conexión predeterminado afecta a todas las etapas de la API. Esta configuración se hace efectiva

cuando actualiza cualquier configuración en cualquier etapa, como al actualizar la implementación en la etapa.

El siguiente procedimiento muestra cómo deshabilitar el punto de conexión predeterminado para una API de REST.

Consola de administración de AWS

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST.
3. En el panel de navegación principal, elija Configuración de la API.
4. Elegir una API.
5. En Detalles de API, elija Editar.
6. En Punto de conexión predeterminado, elija Inactivo.
7. Seleccione Save changes (Guardar cambios).
8. En el panel de navegación principal, elija Recursos.
9. Elija Implementar API.
10. Vuelva a implementar la API en una etapa o actualice cualquier configuración en una etapa para que la actualización surta efecto.

AWS CLI

El siguiente comando update-rest-api permite deshabilitar el punto de conexión predeterminado:

```
aws apigateway update-rest-api \
    --rest-api-id abcdef123 \
    --patch-operations op=replace,path=/disableExecuteApiEndpoint,value='True'
```

Después de desactivar el punto de conexión predeterminado, debe implementar la API para que el cambio surta efecto.

El siguiente comando create-deployment crea una implementación y la asocia a una etapa:

```
aws apigateway create-deployment \
    --rest-api-id abcdef123 \
    --stage-name dev
```

Configuración de las comprobaciones de estado personalizadas para la conmutación por error de DNS para una API de API Gateway

Puede utilizar las comprobaciones de estado de Amazon Route 53 para controlar la conmutación por error de DNS de una API de API Gateway en una Región de AWS principal a una en una región secundaria. Esto puede ayudar a mitigar los impactos en caso de un problema regional. Si utiliza un dominio personalizado, puede realizar una conmutación por error sin necesidad de que los clientes cambien los puntos de conexión de la API.

Al elegir Evaluate Target Health (Evaluar estado del destino) como registro de alias, esos registros solo producen un error cuando el servicio de API Gateway no está disponible en la región. En algunos casos, las propias API de API Gateway pueden sufrir interrupciones antes de ese momento. Para controlar directamente la conmutación por error de DNS, configure comprobaciones de estado personalizadas de Route 53 para las API de API Gateway. Para este ejemplo, se utiliza una alarma de CloudWatch que ayuda a los operadores a controlar la conmutación por error de DNS. Para obtener más ejemplos y otras consideraciones al configurar la conmutación por error, consulte Creación de mecanismos de recuperación de desastres mediante Route 53 y Realización de comprobaciones de estado de Route 53 en los recursos privados de una VPC con AWS Lambda y CloudWatch.

Temas

- Requisitos previos
- Paso 1: Configurar recursos
- Paso 2: Iniciar la conmutación por error a la región secundaria
- Paso 3: Probar la conmutación por error
- Paso 4: Volver a la región principal
- Próximos pasos: Personalizar y probar con regularidad

Requisitos previos

Para completar este procedimiento, debe crear y configurar los siguientes recursos:

- El nombre de un dominio de su propiedad.
- Un certificado de ACM para ese nombre de dominio en dos Regiones de AWS. Para obtener más información, consulte the section called “Requisitos previos de los nombres de dominio personalizados”.

- Una zona alojada de Route 53 para el nombre de dominio. Para obtener más información, consulte Uso de zonas alojadas en la Guía para desarrolladores de Amazon Route 53.

Para obtener más información sobre cómo crear registros de DNS de conmutación por error de Route 53 para los nombres de dominio, consulte Elegir una política de enrutamiento en la guía para desarrolladores de Amazon Route 53. Para obtener más información sobre cómo monitorear una alarma de CloudWatch, consulte Monitoreo de una alarma de CloudWatch en la guía para desarrolladores de Amazon Route 53.

Paso 1: Configurar recursos

En este ejemplo, se crean los siguientes recursos para configurar la conmutación por error de DNS para el nombre de dominio:

- API de API Gateway en dos Regiones de AWS
- Nombres de dominio personalizados de API Gateway con el mismo nombre en dos Regiones de AWS
- Mapeos de API de API Gateway que conectan las API de API Gateway con los nombres de dominio personalizados
- Registros de DNS de conmutación por error de Route 53 para los nombres de dominio
- Una alarma de CloudWatch en la región secundaria
- Una comprobación de estado de Route 53 basada en la alarma de CloudWatch en la región secundaria

En primer lugar, asegúrese de contar con todos los recursos necesarios en las regiones principal y secundaria. La región secundaria debe contener la alarma y la comprobación de estado. De esta forma, no tiene que depender de la región principal para realizar la conmutación por error. Por ejemplo, plantillas de CloudFormation que crean estos recursos, consulte `primary.yaml` y `secondary.yaml`.

> **Important**
>
> Antes de realizar la conmutación por error a la región secundaria, asegúrese de que estén disponibles todos los recursos necesarios. De lo contrario, la API no estará lista para el tráfico de la región secundaria.

Paso 2: Iniciar la conmutación por error a la región secundaria

En el siguiente ejemplo, la región en espera recibe una métrica de CloudWatch e inicia la conmutación por error. Usamos una métrica personalizada que requiere la intervención del operador para iniciar la conmutación por error.

```
aws cloudwatch put-metric-data \
    --metric-name Failover \
    --namespace HealthCheck \
    --unit Count \
    --value 1 \
    --region us-west-1
```

Sustituya los datos de las métricas por los datos correspondientes a la alarma de CloudWatch que haya configurado.

Paso 3: Probar la conmutación por error

Invoque la API y compruebe que recibe una respuesta de la región secundaria. Si usó las plantillas de ejemplo en el paso 1, la respuesta cambia de `{"message": "Hello from the primary Region!"}` a `{"message": "Hello from the secondary Region!"}` después de la conmutación por error.

```
curl https://my-api.example.com

{"message": "Hello from the secondary Region!"}
```

Paso 4: Volver a la región principal

Para volver a la región principal, envíe una métrica de CloudWatch que haga que se apruebe la comprobación de estado.

```
aws cloudwatch put-metric-data \
    --metric-name Failover \
    --namespace HealthCheck \
    --unit Count \
    --value 0 \
    --region us-west-1
```

Sustituya los datos de las métricas por los datos correspondientes a la alarma de CloudWatch que haya configurado.

Invoque la API y compruebe que recibe una respuesta de la región principal. Si usó las plantillas de ejemplo en el paso 1, la respuesta cambia de `{"message": "Hello from the secondary Region!"}` a `{"message": "Hello from the primary Region!"}`.

```
curl https://my-api.example.com

{"message": "Hello from the primary Region!"}
```

Próximos pasos: Personalizar y probar con regularidad

Este ejemplo muestra una forma de configurar la conmutación por error de DNS. Puede utilizar una variedad de métricas o puntos de conexión de HTTP de CloudWatch para las comprobaciones de estado que administran la conmutación por error. Pruebe los mecanismos de conmutación por error con regularidad para asegurarse de que funcionan según lo esperado y de que los operadores están familiarizados con los procedimientos de conmutación por error.

Optimización del rendimiento de las API de REST

Después de que haya hecho que su API esté disponible para ser llamada, es posible que se dé cuenta de que necesita optimizarse para mejorar la capacidad de respuesta. API Gateway proporciona algunas estrategias para optimizar su API, como el almacenamiento en caché de respuesta y la compresión de carga útil. En esta sección, puede aprender a habilitar estas capacidades.

Temas

- Configuración de caché para las API de REST en API Gateway
- Compresión de la carga útil para las API de REST en API Gateway

Configuración de caché para las API de REST en API Gateway

Puede habilitar el almacenamiento en caché de la API en API Gateway para almacenar en caché las respuestas del punto de conexión. Con el almacenamiento en caché, puede reducir el número de llamadas realizadas al punto de conexión y mejorar también la latencia de las solicitudes a la API.

Cuando habilita el almacenamiento en caché para una etapa, API Gateway almacena en caché las respuestas del punto de conexión durante el período de tiempo de vida (TTL) especificado, en segundos. A continuación, API Gateway responde a la solicitud buscando la respuesta del punto

de conexión en la caché en lugar de realizar una solicitud al punto de conexión. El valor de TTL predeterminado para el almacenamiento en caché de la API es de 300 segundos. El valor de TTL máximo es de 3600 segundos. TTL = 0 significa que el almacenamiento en caché está deshabilitado.

> **Note**
>
> El almacenamiento en caché es el mejor esfuerzo. Puede utilizar las métricas `CacheHitCount` y `CacheMissCount` en Amazon CloudWatch para monitorear las solicitudes que API Gateway atiende desde la caché de la API.

El tamaño máximo de una respuesta que se puede almacenar en la caché es 1048576 bytes. El cifrado de datos en la caché puede aumentar el tamaño de la respuesta cuando se almacena en la caché.

Se trata de un servicio compatible con HIPAA. Para obtener más información acerca de AWS, la ley de responsabilidad y portabilidad de seguros médicos de Estados Unidos de 1996 (HIPAA) y el uso de los servicios de AWS para procesar, almacenar y transmitir información sanitaria protegida (PHI), consulte Información general sobre la HIPAA.

> **Important**
>
> Al habilitar el almacenamiento en caché para una etapa, solo los métodos `GET` tienen el almacenamiento en caché habilitado de forma predeterminada. Esto ayuda a garantizar la seguridad y la disponibilidad de la API. Puede habilitar el almacenamiento en caché para otros métodos invalidando la configuración del método.

> **Important**
>
> El almacenamiento en caché se cobra por hora en función del tamaño de la memoria caché que seleccione. El almacenamiento en memoria caché no está disponible para la capa gratuita de AWS. Para obtener más información, consulte Precio de API Gateway.

Activar almacenamiento en caché de Amazon API Gateway

En API Gateway, puede habilitar el almacenamiento en caché para una etapa específica.

Al habilitar el almacenamiento en caché, debe elegir una capacidad de caché. En general, una capacidad mayor ofrece un mejor desempeño, pero también cuesta más. Para conocer los tamaños de la memoria caché admitidos, consulte cacheClusterSize en la referencia de API de API Gateway.

API Gateway permite el almacenamiento en caché creando una instancia de caché dedicada. Este proceso puede tardar hasta cuatro minutos.

API Gateway cambia la capacidad de almacenamiento en caché eliminando al instancia de caché y creando una nueva con una capacidad modificada. Todos los datos almacenados en la memoria caché existente se eliminan.

> **Note**
>
> La capacidad de la caché afecta la CPU, la memoria y el ancho de banda de red de la instancia de caché. Como resultado, la capacidad de la caché puede afectar el rendimiento de dicha caché.
> API Gateway recomienda ejecutar una prueba de carga de 10 minutos para comprobar que la capacidad de la caché sea adecuada para la carga de trabajo. Asegúrese de que durante la prueba de carga el tráfico refleje el tráfico de producción. Por ejemplo, incluya aumento gradual, tráfico constante y picos de tráfico. La prueba de carga debe incluir respuestas que se puedan entregar desde la caché, así como respuestas individuales que agreguen elementos a dicha caché. Monitoree las métricas de latencia, 4xx, 5xx, métricas de aciertos y errores de la caché durante la prueba de carga. Ajuste la capacidad de la caché según sea necesario en función de estas métricas. Para obtener más información sobre las pruebas de carga, consulte ¿Cómo selecciono la mejor capacidad de memoria caché de API Gateway para evitar alcanzar un límite de velocidad?.

Consola de administración de AWS

En la consola de API Gateway, se configura el almacenamiento en caché en la página Etapas. Debe aprovisionar la caché de etapas y especificar una configuración de caché predeterminada en el nivel de método. Si activa la caché en el nivel de método predeterminada, la caché en el nivel de método se activa para todos los métodos `GET` de la etapa, a menos que ese método tenga una invalidación de método. Cualquier método `GET` adicional que implemente en la etapa tendrá una caché en el nivel de método. Para configurar los ajustes de almacenamiento en caché en el nivel de método para métodos específicos de la etapa, puede utilizar invalidaciones de método. Para obtener más información acerca de las invalidaciones de métodos, consulte the

section called “Invalidar el almacenamiento en caché de etapas para el almacenamiento en caché de métodos”.

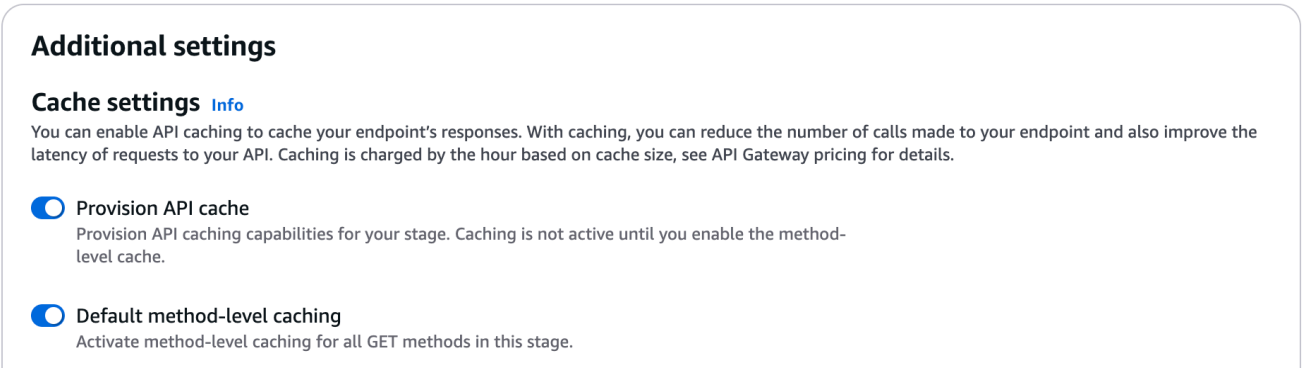
Para configurar el almacenamiento en caché de la API para una determinada etapa:

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Elija Stages (Etapas).
3. En la lista Stages (Etapas) de la API, elija la etapa.
4. En la sección Detalles de la etapa, elija Editar.
5. En Configuración adicional, para Configuración de caché, active Aprovisionar caché de API.

   Esto proporciona un clúster de caché para la etapa.
6. Para activar el almacenamiento en caché para la etapa, active Almacenamiento en caché de nivel de método predeterminado.

   Esto activa el almacenamiento en caché en el nivel de método para todos los métodos `GET` de la etapa. Cualquier método `GET` adicional que implemente en esta etapa tendrá una caché en el nivel de método.

   > **Note**
   >
   > Si ya tiene una configuración para una caché en el nivel de método, el cambio de la configuración predeterminada de almacenamiento en caché en el nivel de método no afectará a esa configuración existente.



7. Seleccione Save changes (Guardar cambios).

AWS CLI

El siguiente comando update-stage permite actualizar una etapa para aprovisionar una caché y activar el almacenamiento en caché de nivel de método para todos los métodos GET de la etapa:

```
aws apigateway update-stage \
    --rest-api-id a1b2c3 \
    --stage-name 'prod' \
    --patch-operations file://patch.json
```

El contenido de `patch.json` es el siguiente:

```
[
     {
          "op": "replace",
          "path": "/cacheClusterEnabled",
          "value": "true"
     },
     {
          "op": "replace",
          "path": "/cacheClusterSize",
          "value": "0.5"
     },
     {
        "op": "replace",
        "path": "/*/*/caching/enabled",
        "value": "true"
     }
]
```

> **Note**
>
> Si ya tiene una configuración para una caché en el nivel de método, el cambio de la configuración predeterminada de almacenamiento en caché en el nivel de método no afectará a esa configuración existente.

> **Note**
>
> La creación o eliminación de una memoria caché tarda unos cuatro minutos en completarse en API Gateway.

Cuando se crea una caché, el valor de Clúster de caché cambia de `Create in progress` a `Active`. Cuando se completa la eliminación de la caché, el valor de Clúster de caché cambia de `Delete in progress` a `Inactive`.
Al activar el almacenamiento en caché en el nivel de método para todos los métodos de la etapa, el valor de Almacenamiento en caché en el nivel de método predeterminado cambia a `Active`. Si desactiva el almacenamiento en caché en el nivel de método para todos los métodos de la etapa, el valor de Almacenamiento en caché en el nivel de método predeterminado cambia a `Inactive`. Si ya tiene una configuración para una caché en el nivel de método, el cambio del estado de la caché no afecta a esa configuración.

Cuando se habilita el almacenamiento en caché en Configuración de caché de la etapa, solo se almacenan en caché los métodos `GET`. Para garantizar la seguridad y la disponibilidad de la API, le recomendamos que no cambie esta configuración. Sin embargo, puede habilitar el almacenamiento en caché para otros métodos invalidando la configuración del método.

Si desea comprobar si el almacenamiento en caché funciona según lo previsto, tiene dos opciones generales:

- Revise las métricas de CloudWatch de CacheHitCount y CacheMissCount para la API y la etapa.
- Inserte una marca de tiempo en la respuesta.

Note

No utilice el encabezado `X-Cache` de la respuesta de CloudFront para determinar si la API se está sirviendo desde la instancia de caché de API Gateway.

Invalidación del almacenamiento en caché en el nivel de etapa de API Gateway para el almacenamiento en caché en el nivel de método

Puede invalidar la configuración de caché en el nivel de etapa activando o desactivando el almacenamiento en caché para un método específico. Puede también modificar el periodo de TTL o activar o desactivar el cifrado para respuestas almacenadas en caché.

Si prevé que un método que está almacenando en caché va a recibir información confidencial en sus respuestas, cifre los datos de la caché. Es probable que deba hacerlo para asegurar el

cumplimiento con diversos marcos normativos. Para obtener más información, consulte Controles de Amazon API Gateway en la Guía del usuario de AWS Security Hub.

Consola de administración de AWS

Si cambia la configuración predeterminada del almacenamiento en caché en el nivel de método en Detalles de la etapa, la configuración de la caché en el nivel de método que tiene invalidaciones no se verá afectada.

Si prevé que un método que almacena en caché va a recibir información confidencial en sus respuestas, en Cache Settings (Configuración de caché), elija Encrypt cache data (Cifrar datos de caché).

Para configurar los métodos individuales de almacenamiento en caché de la API para utilizar la consola:

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija la API.
3. Elija Stages (Etapas).
4. En la lista Stages (Etapas) de la API, expanda la etapa y elija un método en la API.
5. En la sección Invalidaciones de métodos, elija Editar.
6. En la sección Configuración del método, active o desactive Habilitar la caché de métodos o personalice cualquier otra opción que desee.

   > **Note**
   >
   > El almacenamiento en caché no está activo hasta que aprovisione un clúster de caché para la etapa.

7. Seleccione Save.

AWS CLI

El siguiente comando update-stage permite desactivar la caché solo para el método `GET /pets`:

```
aws apigateway update-stage /
    --rest-api-id a1b2c3 /
    --stage-name 'prod' /
```

```
    --patch-operations file://patch.json
```

El contenido de `patch.json` es el siguiente:

```
[{
        "op": "replace",
        "path": "/~1pets/GET/caching/enabled",
        "value": "false"
}]
```

Usar parámetros de método o integración como claves de caché para indexar las respuestas almacenadas en caché

Puede usar un parámetro de método o integración como claves de caché para indexar las respuestas almacenadas en caché. Esto incluye encabezados personalizados, rutas de URL o cadenas de consulta. Puede especificar alguno de estos parámetros o todos ellos como la clave de caché, pero debe especificar al menos un valor. Cuando tiene una clave de caché, API Gateway almacena en caché las respuestas de cada valor de clave de forma independiente, incluso cuando la clave de caché no está presente.

> **Note**
>
> Las claves de caché son necesarias cuando se configura el almacenamiento en caché en un recurso.

Suponga, por ejemplo, que tiene una solicitud con el siguiente formato:

```
GET /users?type=... HTTP/1.1
host: example.com
...
```

En esta solicitud, `type` puede tomar un valor de `admin` o `regular`. Si incluye el parámetro `type` como parte de la clave de caché, las respuestas de `GET /users?type=admin` se almacenan en caché por separado de las de `GET /users?type=regular`.

Cuando una solicitud de método o integración toma más de un parámetro, puede elegir algunos o todos los parámetros para crear la clave de caché. Por ejemplo, puede incluir solamente el parámetro `type` en la clave de caché para la siguiente solicitud, realizada en el orden indicado dentro de un período de TTL:

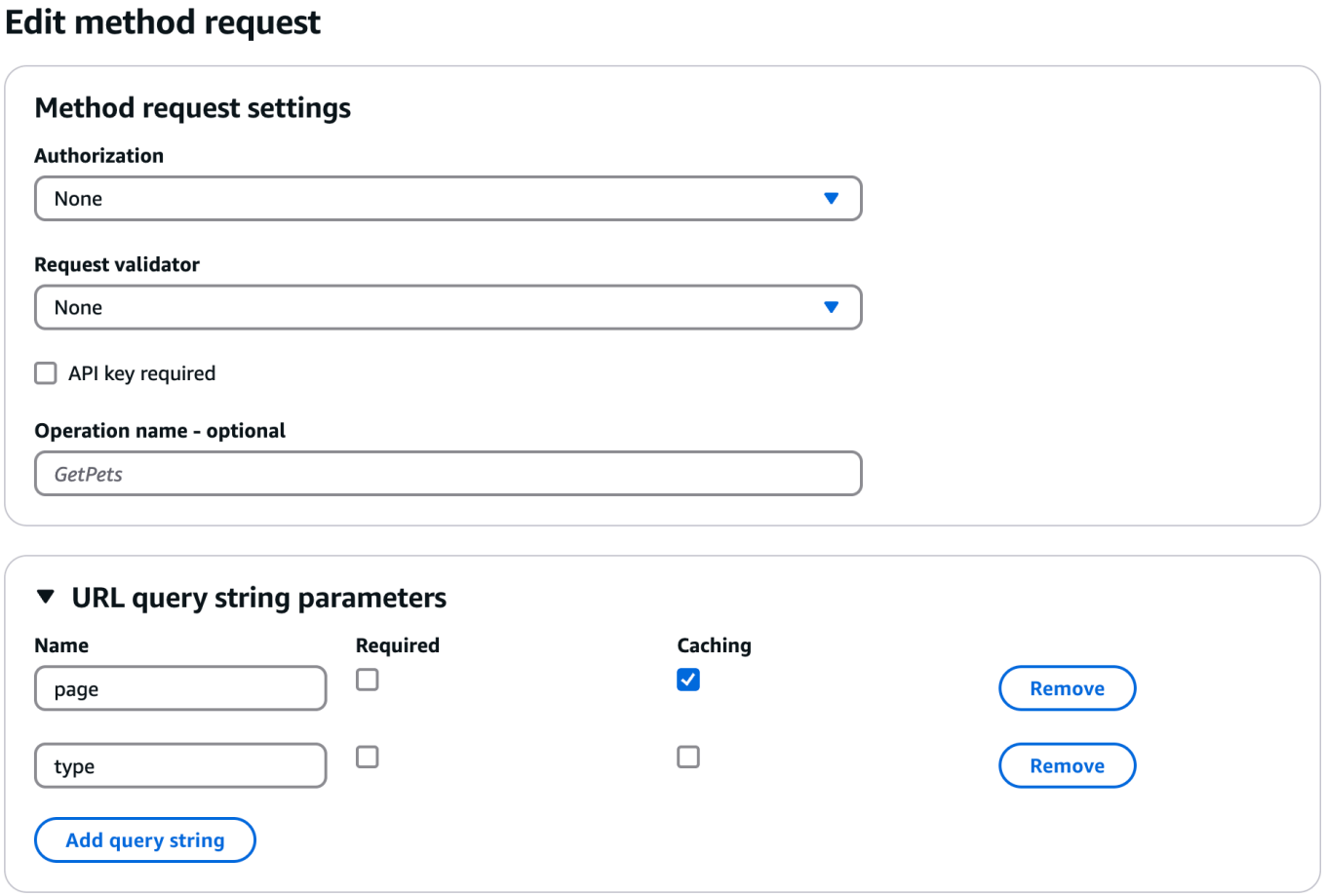
```
GET /users?type=admin&department=A HTTP/1.1
host: example.com
...
```

La respuesta de esta solicitud se almacena en caché y se utiliza para servir la siguiente solicitud:

```
GET /users?type=admin&department=B HTTP/1.1
host: example.com
...
```

Consola de administración de AWS

Para incluir un parámetro de solicitud de método o integración como parte de una clave de caché en la consola de API Gateway, seleccione Caching después de agregar el parámetro.



AWS CLI

El siguiente comando put-method permite crear un método GET y requiere el parámetro de cadena de consulta `type`:

```
aws apigateway put-method /
    --rest-api-id a1b2c3 /
    --resource-id aaa111 /
    --http-method GET /
    --authorization-type "NONE" /
    --request-parameters "method.request.querystring.type=true"
```

El siguiente comando put-integration permite crear una integración para el método GET con un punto de conexión HTTP y especifica que API Gateway almacene en caché el parámetro de solicitud del método `type`:

```
aws apigateway put-integration /
    --rest-api-id a1b2c3 /
```

```
    --resource-id aaa111 /
    --http-method GET /
    --type HTTP /
    --integration-http-method GET /
    --uri 'https://example.com' /
    --cache-key-parameters "method.request.querystring.type"
```

Para especificar que API Gateway almacene en caché un parámetro de solicitud de integración, utilice `integration.request.location.name` como parámetro de clave de caché.

Vaciar la caché de etapa de API en API Gateway

Cuando el almacenamiento en caché de la API está habilitado, puede vaciar la caché de etapas de API para asegurarse de que los clientes de la API obtengan las respuestas más recientes de los puntos de conexión de integración.

Consola de administración de AWS

Vaciado de la caché de etapas de la API

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Elija una API que tenga una etapa con una caché.
3. En el panel de navegación principal, elija Etapas y, a continuación, elija una etapa con caché.
4. Elija el menú Acciones de etapa y, a continuación, seleccione Vaciar la caché de etapas.

AWS CLI

El siguiente comando flush-stage-cache permite vaciar la caché de etapas:

```
aws apigateway flush-stage-cache \
    --rest-api-id a1b2c3 \
    --stage-name prod
```

> **Note**
>
> Después de que la caché se vacía, las respuestas se atienden desde el punto de conexión de integración hasta que se vuelva a crear la caché. Durante este período, el número de

solicitudes enviadas al punto de conexión de integración puede aumentar. Esto podría aumentar temporalmente la latencia total de la API.

Invalidar una entrada de caché de API Gateway

Un cliente de la API puede invalidar una entrada de caché existente y volver a cargarla desde el punto de conexión de integración para las distintas solicitudes. El cliente debe enviar una solicitud que contenga el encabezado `Cache-Control: max-age=0`. El cliente recibe la respuesta directamente del punto de conexión de integración en lugar de la caché, siempre que el cliente esté autorizado para ello. Esto sustituye la entrada de caché existente por la nueva respuesta, que se obtiene del punto de conexión de integración.

Para conceder permiso a un cliente, asocie una política con el siguiente formato a una función de ejecución de IAM del usuario.

> **Note**
>
> No se admite la invalidación de la memoria caché entre cuentas.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "execute-api:InvalidateCache"
            ],
            "Resource": [
                "arn:aws:execute-api:us-east-1:111111111111:api-id/stage-name/
GET/resource-path-specifier"
            ]
        }
    ]
}
```

Esta política permite al servicio de ejecución de API Gateway invalidar la caché para las solicitudes en el recurso o recursos especificados. Para especificar un grupo de recursos de destino, utilice el carácter comodín (*) para `account-id`, `api-id` y otras entradas en el valor de ARN de `Resource`. Para obtener más información sobre cómo establecer permisos para el servicio de ejecución de API Gateway, consulte Control del acceso a una API de REST con permisos de IAM.

Si no impone una política `InvalidateCache` (o selecciona la casilla de verificación Require authorization (Solicitar autorización)), cualquier cliente puede invalidar la caché de la API. Si la mayoría o todos los clientes invalidan la caché de la API, esto podría aumentar considerablemente la latencia de la API.

Cuando la política está implantada, se habilita el almacenamiento en caché y se requiere autorización.

Para especificar el modo en que API Gateway gestiona las solicitudes no autorizadas, seleccione una de las opciones siguientes:

Error de la solicitud con código de estado 403

API Gateway devuelve una respuesta `403 Unauthorized`.

Para establecer esta opción mediante la API, use `FAIL_WITH_403`.

Omisión del encabezado de control de caché; adición de una advertencia en el encabezado de la respuesta

API Gateway procesa la solicitud y agrega un encabezado de advertencia en la respuesta.

Para establecer esta opción mediante la API, use `SUCCEED_WITH_RESPONSE_HEADER`.

Omisión del encabezado de control de caché

API Gateway procesa la solicitud y no agrega un encabezado de advertencia en la respuesta.

Para establecer esta opción mediante la API, use `SUCCEED_WITHOUT_RESPONSE_HEADER`.

Puede establecer el comportamiento de gestión de las solicitudes no autorizadas con la consola de API Gateway o con AWS CLI.

Consola de administración de AWS

Para especificar cómo se gestionan las solicitudes no autorizadas

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Elija una API que tenga una etapa con una caché.
3. En el panel de navegación principal, elija Etapas y, a continuación, elija una etapa con caché.
4. Para ver los Detalles de la etapa, seleccione Editar.
5. Seleccione una opción para Gestión de solicitudes no autorizadas.
6. Elija Continuar.
7. Revise los cambios y elija Guardar cambios.

AWS CLI

El siguiente comando update-stage actualiza una etapa para gestionar las solicitudes no autorizadas devolviendo un error con código de estado 403 para la solicitud:

```
aws apigateway update-stage /
    --rest-api-id a1b2c3 /
    --stage-name 'prod' /
    --patch-operations 'op=replace,path=/*/*/caching/
unauthorizedCacheControlHeaderStrategy,value="FAIL_WITH_403"'
```

CloudFormationEjemplo de de una etapa con caché

La siguiente plantilla de CloudFormation crea una API de ejemplo, aprovisiona una caché de `0.5` GB para la etapa `Prod` y activa el almacenamiento en caché de nivel de método para todos los métodos `GET`.

> **Important**
>
> El almacenamiento en caché se cobra por hora en función del tamaño de la memoria caché que seleccione. El almacenamiento en memoria caché no está disponible para la capa gratuita de AWS. Para obtener más información, consulte Precio de API Gateway.

Plantilla de CloudFormation de ejemplo

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Name: cache-example
  PetsResource:
    Type: 'AWS::ApiGateway::Resource'
    Properties:
      RestApiId: !Ref Api
      ParentId: !GetAtt Api.RootResourceId
      PathPart: 'pets'
  PetsMethodGet:
    Type: 'AWS::ApiGateway::Method'
    Properties:
      RestApiId: !Ref Api
      ResourceId: !Ref PetsResource
      HttpMethod: GET
      ApiKeyRequired: true
      AuthorizationType: NONE
      Integration:
        Type: HTTP_PROXY
        IntegrationHttpMethod: GET
        Uri: http://petstore-demo-endpoint.execute-api.com/petstore/pets/
  ApiDeployment:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn:
      - PetsMethodGet
    Properties:
      RestApiId: !Ref Api
  ApiStage:
    Type: 'AWS::ApiGateway::Stage'
    Properties:
      StageName: Prod
      Description: Prod Stage with a cache
      RestApiId: !Ref Api
      DeploymentId: !Ref ApiDeployment
      CacheClusterEnabled: True
      CacheClusterSize: 0.5
      MethodSettings:
        - ResourcePath: /*
          HttpMethod: '*'
```

```
            CachingEnabled: True
```

Compresión de la carga útil para las API de REST en API Gateway

API Gateway permite que el cliente llame a la API con cargas comprimidas utilizando una de las codificaciones de contenido compatibles. De forma predeterminada, API Gateway admite la descompresión de la carga de solicitud del método. Sin embargo, debe configurar la API para que se habilite la compresión de la carga de respuesta del método.

Para habilitar la compresión de una `API`, establezca la propiedad `minimumCompressionsSize` en un número entero que no sea negativo y esté comprendido entre 0 y 10485760 (10 M bytes) cuando cree la API o después de crearla. Para deshabilitar la compresión de la API, establezca `minimumCompressionSize` en null o elimínelo por completo. Puede habilitar o desactivar la compresión de una API mediante la consola de API Gateway, la AWS CLI o la API REST de API Gateway.

Si desea que la compresión se aplique en una carga de cualquier tamaño, establezca el valor `minimumCompressionSize` en cero. Sin embargo, es posible que los datos pequeños, al comprimirse, aumenten de tamaño. Además, la compresión de API Gateway y la descompresión en el cliente podrían aumentar la latencia general y requerir más tiempo de procesamiento. Debe ejecutar los casos de prueba en la API para determinar un valor óptimo.

El cliente puede enviar una solicitud de API con una carga comprimida y un encabezado `Content-Encoding` adecuado para que API Gateway descomprima y aplique las plantillas de asignaciones correspondientes antes de pasar la solicitud al punto de conexión de integración. Una vez que la compresión está habilitada y la API está implementada, el cliente puede recibir una respuesta de API con una carga comprimida si especifica un encabezado `Accept-Encoding` adecuado en la solicitud del método.

Si el punto de conexión de integración espera y devuelve cargas JSON sin comprimir, las plantillas de asignación configuradas para una carga JSON sin comprimir serán aplicables a la carga comprimida. En el caso de las cargas de solicitud de métodos comprimidas, API Gateway las descomprime, aplica la plantilla de asignación y pasa la solicitud comprimida al punto de conexión de integración. En el caso de las cargas de respuesta de integración sin comprimir, API Gateway aplica la plantilla de asignación, comprime la carga asignada y devuelve la carga comprimida al cliente.

Temas

- Habilitación de la compresión de la carga para una API en API Gateway
- Llamada a un método de API con una carga comprimida en API Gateway

- Recepción de una respuesta de API con una carga comprimida en API Gateway

Habilitación de la compresión de la carga para una API en API Gateway

Puede habilitar la compresión de una API mediante la consola de API Gateway, la AWS CLI o un AWS SDK.

En el caso de una API existente, debe implementar la API después de habilitar la compresión para que el cambio surta efecto. Para una nueva API, puede implementar la API después de que se haya completado la configuración de la API.

> **Note**
>
> La codificación de contenido de mayor prioridad debe ser una compatible con la API Gateway. Si no es así, la compresión no se aplica a la carga de la respuesta.

Temas

- Habilitación de la compresión de la carga en una API a través de la consola de API Gateway
- Habilitación de la compresión de carga en una API a través de la AWS CLI
- Codificaciones de contenido admitidas por API Gateway

Habilitación de la compresión de la carga en una API a través de la consola de API Gateway

En el siguiente procedimiento, se describe cómo habilitar la compresión de la carga en una API.

Para habilitar la compresión de carga a través de la consola de API Gateway

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Seleccione una API existente o cree una nueva.
3. En el panel de navegación principal, elija Configuración de la API.
4. En la sección Detalles de la API, elija Editar.
5. Active la codificación de contenido para habilitar la compresión de carga. En Tamaño mínimo del cuerpo, ingrese un número para el tamaño mínimo de compresión (en bytes). Para desactivar la compresión, desactive la opción codificación de contenido.
6. Seleccione Save changes (Guardar cambios).

Habilitación de la compresión de carga en una API a través de la AWS CLI

El siguiente comando create-rest-api permite crear una API con compresión de la carga útil:

```
aws apigateway create-rest-api \
    --name "My test API" \
    --minimum-compression-size 0
```

El siguiente comando update-rest-api permite habilitar la compresión de la carga útil para una API existente:

```
aws apigateway update-rest-api \
    --rest-api-id 1234567890 \
    --patch-operations op=replace,path=/minimumCompressionSize,value=0
```

La propiedad `minimumCompressionSize` presenta un valor entero no negativo entre 0 y 10485760 (10 megabytes). Mide el umbral de compresión. Si el tamaño de la carga es menor que este valor, la compresión o la descompresión no se aplican en la carga. Si se establece en cero, la compresión está permitida para cualquier tamaño de carga.

El siguiente comando update-rest-api permite desactivar la compresión de la carga útil:

```
aws apigateway update-rest-api \
    --rest-api-id 1234567890 \
    --patch-operations op=replace,path=/minimumCompressionSize,value=
```

También puede definir `value` en una cadena vacía `""` u omitir la propiedad `value` en su conjunto en la llamada anterior.

Codificaciones de contenido admitidas por API Gateway

API Gateway admite las siguientes codificaciones de código:

- `deflate`
- `gzip`
- `identity`

API Gateway también admite el siguiente formato de encabezado `Accept-Encoding`, conforme a la especificación 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`

Llamada a un método de API con una carga comprimida en API Gateway

Para realizar una solicitud de API con una carga comprimida, el cliente debe configurar el encabezado `Content-Encoding` utilizando una de las codificaciones de contenido admitidas.

Supongamos que es un cliente de API y que quiere llamar al método PetStore (`POST /pets`). No llame al método utilizando la siguiente salida JSON:

```
POST /pets
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
Content-Length: ...

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

En su lugar, puede llamar al método con la misma carga comprimida utilizando la codificación GZIP:

```
POST /pets
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
Content-Encoding:gzip
Content-Length: ...

[illegible]RPP*[illegible],HU[illegible]RPJ[illegible]OW[illegible]e&[illegible]L,[illegible],-y[illegible]j
```

Cuando API Gateway recibe la solicitud, comprueba si la codificación de contenido especificada es compatible. A continuación, intenta descomprimir la carga con la codificación de contenido especificada. Si la descompresión se realiza correctamente, envía la solicitud al punto de conexión de integración. Si la codificación especificada no es compatible o la carga suministrada no está comprimida con la codificación especificada, API Gateway devuelve la respuesta de error `415 Unsupported Media Type`. El error no se registra en CloudWatch Logs, si se produce en las fases iniciales de descompresión antes de que se identifiquen la API y la etapa.

Recepción de una respuesta de API con una carga comprimida en API Gateway

Cuando realiza una solicitud en una API habilitada para la compresión, el cliente puede optar por recibir una carga de respuesta comprimida de un formato específico especificando un encabezado `Accept-Encoding` con una codificación de contenido compatible.

API Gateway solo comprime la carga de respuesta cuando se cumplen las siguientes condiciones:

- La solicitud entrante tiene el encabezado `Accept-Encoding` con el formato y la codificación de contenido compatibles.

  > **Note**
  >
  > Si no se define el encabezado, el valor predeterminado es * tal y como se define en RFC 7231. En tal caso, API Gateway no comprime la carga útil. Algunos navegadores o clientes pueden añadir `Accept-Encoding` (por ejemplo `Accept-Encoding:gzip, deflate, br`) automáticamente para las solicitudes habilitadas para compresión. Esto puede activar la compresión de la carga útil en API Gateway. Sin una especificación explícita de los valores del encabezado `Accept-Encoding` admitidos, API Gateway no comprime la carga.

- `minimumCompressionSize` está establecido en la API para habilitar la compresión.
- La respuesta de integración no tiene un encabezado `Content-Encoding`.
- El tamaño de la carga de respuesta de integración, una vez que se aplica la plantilla de asignación correspondiente, es mayor o igual que el valor `minimumCompressionSize` especificado.

API Gateway aplica una plantilla de asignación configurada para la respuesta de integración antes de que se comprima la carga. Si la respuesta de integración contiene un encabezado `Content-Encoding`, API Gateway presupone que la carga de respuesta de integración ya está comprimida y omite el proceso de compresión.

Un ejemplo de esto es el de la API de PetStore y la siguiente solicitud:

```
GET /pets
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
Accept: application/json
```

El backend responde a la solicitud con una carga JSON sin comprimir similar a la siguiente:

```
200 OK

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

Para recibir esta salida como una carga comprimida, el cliente de la API puede enviar una solicitud como la siguiente:

```
GET /pets
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
Accept-Encoding:gzip
```

El cliente recibe la respuesta con un encabezado `Content-Encoding` y una carga codificada en GZIP similar a la siguiente:

```
200 OK
Content-Encoding:gzip
...

���RP�

J�)JV
�:P^IeA*������+(�L ��X�YZ�ku0L0B7!9��C#�&����Y��a���^�X
```

Cuando la carga de respuesta está comprimida, solo el tamaño de los datos comprimidos se tiene en cuenta al facturar la transferencia de información.

Distribución de sus API de REST a los clientes en API Gateway

En esta sección se proporciona información detallada sobre la distribución de API Gateway a sus clientes. Distribuir la API incluye generar SDK para que los clientes los descarguen e integren con las aplicaciones cliente, documentar la API para que los clientes sepan cómo llamarla desde las aplicaciones cliente y hacer que la API esté disponible como parte de las ofertas de productos.

Temas

- Planes de uso y clave de API para las API de REST en API Gateway
- Documentación para las API de REST en API Gateway
- Generación de SDK para las API de REST en API Gateway
- Venda sus API de API Gateway a través de AWS Marketplace

Planes de uso y clave de API para las API de REST en API Gateway

Después de crear, probar e implementar las API, puede utilizar los planes de uso de API Gateway para ponerlos a disposición de sus clientes como ofertas de productos. Puede configurar planes de uso y claves de API para permitir que los clientes accedan a las API seleccionadas, y comenzar la limitación controlada de las solicitudes a esas API en función de los límites y cuotas definidos. Se pueden configurar a nivel de API o método API.

¿Qué son los planes de uso y las claves de API?

Un plan de uso especifica quién puede acceder a una o más etapas y métodos de la API implementada, y opcionalmente establece el ratio de solicitudes objetivo para comenzar la limitación controlada de las solicitudes. El plan utiliza claves de API para identificar los clientes de API y a quienes pueden acceder a las etapas de la API asociadas a cada clave.

Las claves de API son valores de cadenas alfanuméricas que distribuye a los clientes desarrolladores de aplicaciones para concederles acceso a su API. Puede utilizar claves de API junto con autorizadores de Lambda, roles de IAM o Amazon Cognito para controlar el acceso a sus API. API Gateway puede generar claves de API en su nombre o usted puede importarlas desde un archivo CSV. Puede generar una clave de API en API Gateway o importarla a API Gateway desde un origen externo. Para obtener más información, consulte the section called “Configuración de claves API para API de REST en API Gateway”.

Una clave de API tiene un nombre y un valor. (Los términos "clave de API" y "valor de clave de API" suelen utilizarse indistintamente). El nombre no puede superar los 1024 caracteres. El valor es una cadena alfanumérica con un tamaño entre 20 y 128 caracteres, por ejemplo, `apikey1234abcdefghij0123456789`.

> **Important**
>
> Los valores de claves de API deben ser únicos. Si intenta crear dos claves de API con nombres distintos pero con el mismo valor, API Gateway considerará que son la misma clave de API.
> Una clave de API se puede asociar a más de un plan de uso. Un plan de uso se puede asociar a más de una etapa. Sin embargo, una clave de API determinada solo se puede asociar a un plan de uso para cada etapa de la API.

Un límite de limitación controlada establece el punto objetivo en el que debe comenzar la limitación controlada de solicitudes. Esto se puede configurar a nivel de API o de método de API.

Un límite de cuota establece el número máximo de solicitudes con una clave de API determinada que se pueden enviar en un intervalo de tiempo especificado. Puede configurar distintos métodos de la API para exigir la autorización de la clave de API en función de la configuración del plan de uso.

La limitación de solicitudes y los límites de cuota se aplican a las solicitudes de claves de API individuales que se van acumulando en todas las etapas de la API dentro de un plan de uso.

> **Note**
>
> La limitación controlada del plan de uso y las cuotas no son límites estrictos y se aplican en la medida de lo posible. En algunos casos, los clientes pueden sobrepasar las cuotas establecidas. Para controlar los costes o bloquear el acceso a una API, no confíe en las cuotas del plan de uso ni en la limitación controlada. Considerar la posibilidad de utilizar AWS Budgets para monitorear los costes y AWS WAF para administrar las solicitudes de API.

Prácticas recomendadas para claves de API y planes de uso

A continuación, se sugieren prácticas recomendadas cuando se utilizan claves de API y planes de uso.

> **Important**
>
> - No utilice claves de API para la autenticación o la autorización del control de acceso a las API. Si tiene varias API en un plan de uso, un usuario con una clave de API válida para una API en ese plan de uso puede acceder a todas las API de ese plan de uso. En su lugar, para controlar el acceso a su API, utilice un rol de IAM, un autorizador de Lambda o un grupo de usuarios de Amazon Cognito.
> - Use las claves de API que genera la puerta de enlace de la API. Las claves de API no deben incluir información confidencial; los clientes suelen transmitirlas en encabezados que se pueden registrar.

- Si está utilizando un portal para desarrolladores para publicar las API, tenga en cuenta que los clientes se pueden suscribir a todas las API de un determinado plan de uso, incluso si no ha hecho que sean visibles para los clientes.
- En algunos casos, los clientes pueden sobrepasar las cuotas establecidas. Para controlar los costos, no confíe en los planes de uso. Considere la posibilidad de utilizar AWS Budgets para monitorear los costos y AWS WAF para administrar las solicitudes de API.
- Después de agregar una clave de API a un plan de uso, es posible que la operación de actualización tarde unos minutos en completarse.

Elección de un origen de clave de API en API Gateway

Al asociar un plan de uso a una API y habilitar claves de API en los métodos de API, cada solicitud de entrada para la API debe contener una clave de API. API Gateway lee la clave y la compara con las claves del plan de uso. Si hay una coincidencia, API Gateway limita las solicitudes en función del límite de solicitudes y la cuota del plan. De lo contrario, inicia una excepción `InvalidKeyParameter`. Como resultado, el intermediario recibe una respuesta `403 Forbidden`.

Su API de API Gateway puede recibir claves de API de una de dos fuentes:

`HEADER`

Distribuye las claves de API a sus clientes y les pide que pasen la clave de API como el encabezado `X-API-Key` de cada solicitud de entrada.

`AUTHORIZER`

Hace que un autorizador de Lambda devuelva la clave de API como parte de la respuesta de autorización. Para obtener más información sobre la respuesta de autorización, consulte the section called “Salida de un autorizador de Lambda de API Gateway”.

> **Note**
>
> Para conocer las prácticas recomendadas a tener en cuenta, consulte the section called “Prácticas recomendadas para claves de API y planes de uso”.

En el procedimiento siguiente muestra cómo elegir un origen de clave de API para una API.

Consola de administración de AWS

Elección de un origen de claves para una API

1. Inicie sesión en la consola de API Gateway.
2. Seleccione una API existente o cree una nueva.
3. En el panel de navegación principal, elija Configuración de la API.
4. En la sección Detalles de la API, elija Editar.
5. En Origen de clave de API, seleccione `Header` o `Authorizer` de la lista desplegable.
6. Seleccione Save changes (Guardar cambios).

AWS CLI

El siguiente comando update-rest-api permite actualizar una API para establecer el origen de clave de API en `AUTHORIZER`:

```
aws apigateway update-rest-api --rest-api-id 1234123412 --patch-operations
 op=replace,path=/apiKeySource,value=AUTHORIZER
```

Para que el cliente envíe una clave de API, establezca el `value` en `HEADER` en el comando anterior.

REST API

Si desea elegir un origen de clave de API a través de la API de REST de API Gateway, llame a `restapi:update`, tal y como se indica a continuación:

```
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"
    }
  ]
}
```

Para que el autorizador devuelva una clave de API, establezca `value` en `AUTHORIZER` en la entrada `patchOperations` anterior.

Formato de archivo de clave de API de API Gateway

API Gateway puede importar claves de API de archivos externos con un formato de valores separados por comas (CSV) y asociar las claves importadas a uno o varios planes de uso. El archivo importado debe contener las columnas `Name` y `Key`. Los nombres de encabezado de columna no distinguen entre mayúsculas y minúsculas y las columnas pueden estar en cualquier orden, tal y como se muestra en el ejemplo siguiente:

```
Key,name
apikey1234abcdefghij0123456789,MyFirstApiKey
```

Un valor `Key` debe estar comprendido entre 20 y 128 caracteres. Un valor `Name` no puede superar los 1024 caracteres.

Un archivo de claves de API puede tener las columnas `Description`, `Enabledo` `UsagePlanIds`, como se muestra en el ejemplo siguiente:

```
Name,key,description,Enabled,usageplanIds
MyFirstApiKey,apikey1234abcdefghij0123456789,An imported key,TRUE,c7y23b
```

Cuando una clave está asociada a varios planes de uso, el valor de `UsagePlanIds` es una cadena separada por comas con los identificadores del plan de uso incluidos entre comillas simples o dobles, tal y como se muestra en el ejemplo siguiente:

```
Enabled,Name,key,UsageplanIds
true,MyFirstApiKey,apikey1234abcdefghij0123456789,"c7y23b,glvrsr"
```

Se pueden usar columnas no reconocidas, pero se omitirán. El valor predeterminado es una cadena vacía o un valor booleano `true`.

Se puede importar varias veces la misma clave de API, pero la versión más reciente sobrescribirá la anterior. Dos claves de API son idénticas si tienen el mismo valor de `key`.

> **Note**
>
> Para conocer las prácticas recomendadas a tener en cuenta, consulte the section called "Prácticas recomendadas para claves de API y planes de uso".

Configuración de claves API para API de REST en API Gateway

Para configurar las claves de API, haga lo siguiente:

- Configure los métodos de la API para que exijan una clave de API.
- Cree o importe una clave de API para la API de una región.

Antes de configurar las claves de API, debe haber creado una API y haberla implementado hasta una fase. Después de crear un valor de clave de API, no se puede cambiar.

Para obtener instrucciones acerca de cómo crear e implementar una API a través de la consola de API Gateway, consulte Desarrollo de las API de REST en API Gateway y Implementación de las API de REST en API Gateway, respectivamente.

Después de crear una clave de API, debe asociarla al plan de uso. Para obtener más información, consulte Configuración de planes de uso para API de REST en API Gateway.

> **Note**
>
> Para conocer las prácticas recomendadas a tener en cuenta, consulte the section called "Prácticas recomendadas para claves de API y planes de uso".

Temas

- Exigir una clave de API en un método
- Crear una clave de API
- Importar claves de API

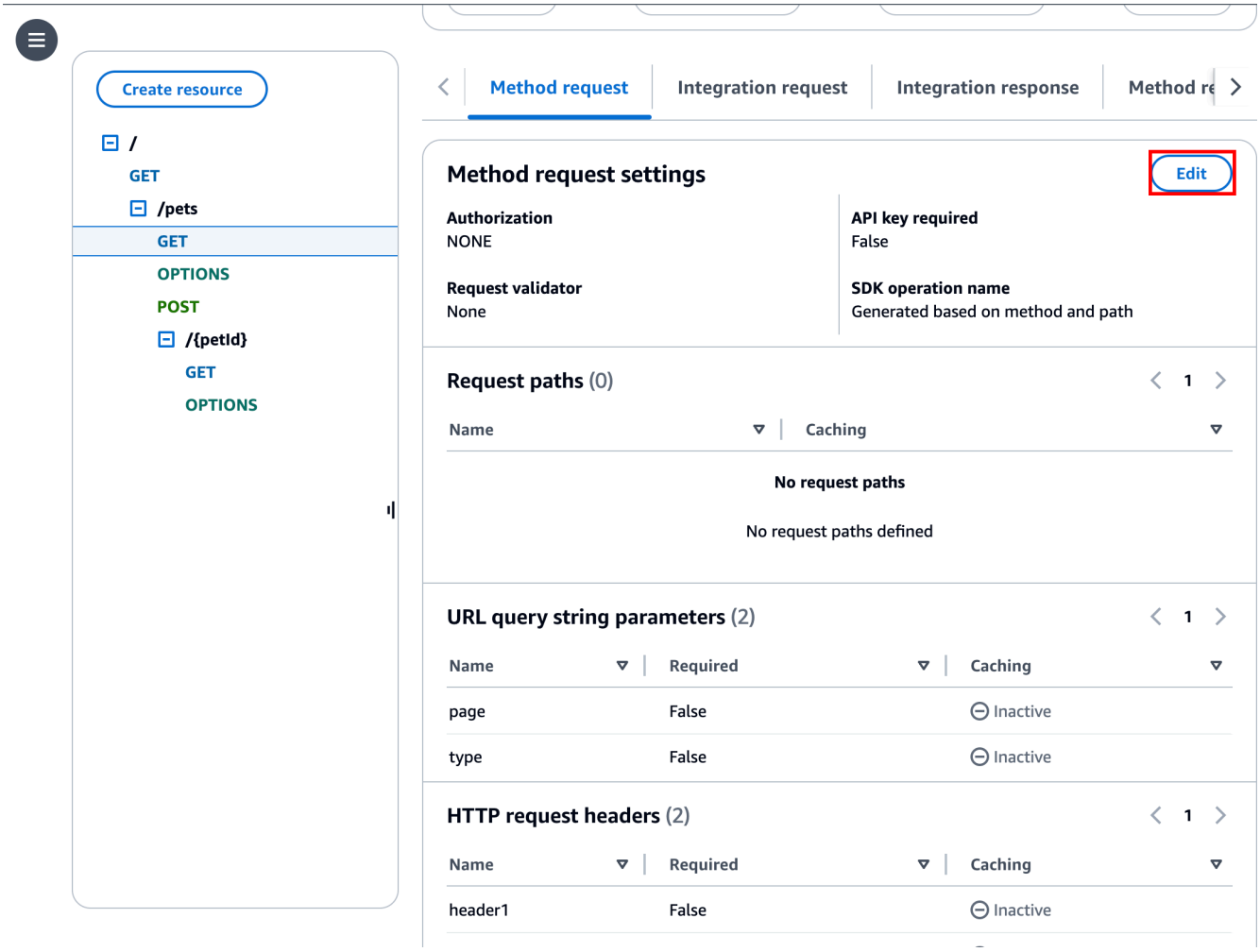
Exigir una clave de API en un método

El siguiente procedimiento describe cómo configurar un método de API para exigir una clave de API.

Consola de administración de AWS

Para configurar un método de API para exigir una clave de API

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST.
3. En el panel de navegación principal de API Gateway, elija Resources (Recursos).
4. En Resources (Recursos), cree un método nuevo o elija uno existente.
5. En la pestaña Solicitud de método, en Configuración de solicitud de método, elija Editar.



6. Seleccione Clave de API obligatoria.
7. Seleccione Save.
8. Implemente o redistribuya la API para que el requisito surta efecto.

Si la opción Clave de API obligatoria está establecida en `false` y no realiza los pasos anteriores, las claves de API asociadas a una etapa de API no se usarán con el método.

AWS CLI

El siguiente comando put-method crea un método PUT que requiere una clave de API:

```
aws apigateway put-method \
    --rest-api-id 1234123412 \
    --resource-id a1b2c3 \
    --http-method PUT \
    --authorization-type "NONE" \
```

```
    --api-key-required
```

El siguiente comando update-method actualiza un método existente para que requiera una clave de API:

```
aws apigateway update-method \
    --rest-api-id 1234123412 \
    --resource-id a1b2c3 \
    --http-method PUT \
    --patch-operations op="replace",path="/apiKeyRequired",value="true"
```

REST API

Para exigir una clave de API en un método, realice alguna de las siguientes operaciones:

- Llame a `method:put` para crear un método. Establezca `apiKeyRequired` en `true` en la carga de la solicitud.
- Llame a `method:update` para establecer `apiKeyRequired` en `true`.

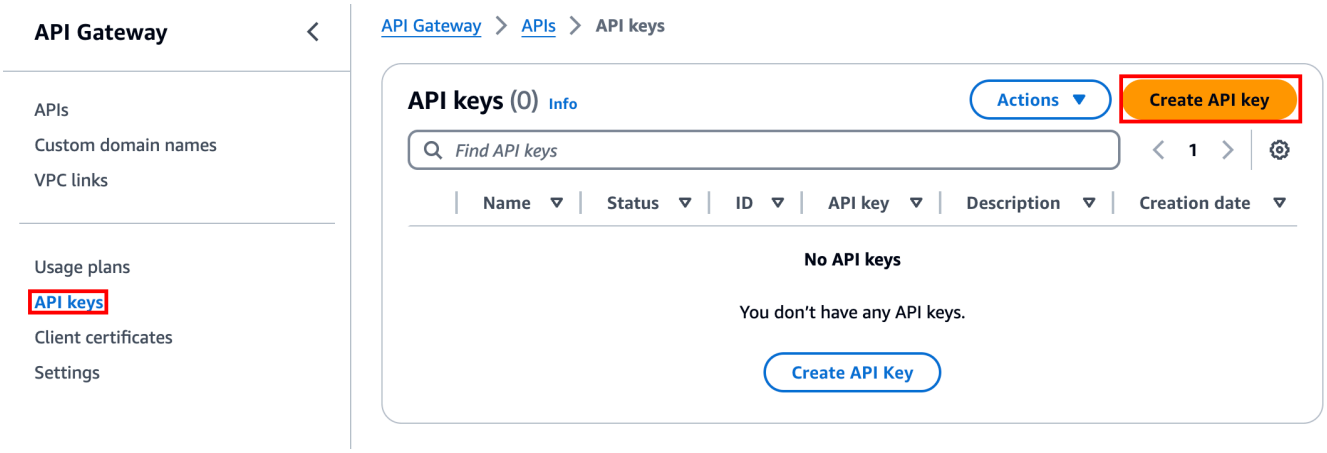
Crear una clave de API

El procedimiento siguiente muestra cómo crear una clave de API. Si desea importar la clave de API, omita este paso.

Consola de administración de AWS

Para crear una clave de API

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST.
3. En el panel de navegación principal de API Gateway, elija Claves de API.
4. Elija Crear clave de la API.



5. En Nombre, ingrese un nombre.
6. (Opcional) En Description (Descripción), introduzca una descripción.
7. En Clave de API, elija Generar automáticamente para que API Gateway genere el valor de la clave o elija Personalizar para crear su propio valor de la clave.
8. Seleccione Save.

AWS CLI

El siguiente comando create-api-key crea una clave de API:

```
aws apigateway create-api-key \
   --name 'Dev API key' \
   --description 'API key for Devs' \
   --enabled
```

REST API

Llame a `apikey:create` para crear una clave de API.

Importar claves de API

En el procedimiento siguiente se describe cómo importar claves de API. Si ya ha creado una clave de API, omita este paso.

Consola de administración de AWS

Para importar claves de API

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST.
3. En el panel de navegación principal, elija Claves de API.
4. Elija el menú desplegable Acciones y, a continuación, elija Importar claves de API.
5. Para cargar un archivo de claves separado por comas, elija Elegir archivo. También puede ingresar las claves en el editor de texto. Para obtener información sobre el formato de los archivos, consulte the section called “Formato de archivo de clave de API de API Gateway”.
6. Elija Error en advertencias para que la importación se detenga cuando se produzca un error o elija Omitir advertencias para que se sigan importando las entradas de clave válidas cuando se produzca una advertencia.
7. Elija Importar para importar las claves de API.

AWS CLI

El siguiente comando import-api-keys importa una clave de API:

```
aws apigateway import-api-key \
    a--body fileb://keys.csv \
    --format csv
```

REST API

Llame a `apikey:import` para importar una clave de API desde un archivo. Para el formato de archivo, consulte Formato de archivo de clave de API de API Gateway.

No se puede cambiar el valor de la nueva clave de API. Después de crear la API, configure un plan de uso. Para obtener más información, consulte Configuración de planes de uso para API de REST en API Gateway.

Configuración de planes de uso para API de REST en API Gateway

Antes de crear un plan de uso, asegúrese de que ha configurado las claves de API que desea. Para obtener más información, consulte Configuración de claves API para API de REST en API Gateway.

Temas

- Migrar la API a planes de uso predeterminados (si es necesario)
- Crear un plan de uso
- Adición de una etapa a un plan de uso
- Adición de una clave API a un plan de uso

Migrar la API a planes de uso predeterminados (si es necesario)

Si comenzó a utilizar API Gateway después del 11 de agosto de 2016 (fecha en la que se implementó la característica de planes de uso), los planes de uso estarán habilitados de forma predeterminada en todas las regiones compatibles.

Si comenzó a utilizar API Gateway antes de dicha fecha, es posible que tenga que migrar a planes de uso predeterminados. Se le solicitará la opción Enable Usage Plans (Habilitar planes de uso) antes de utilizar los planes de uso por primera vez en la región seleccionada. Cuando habilite esta opción, dispondrá de planes de uso predeterminados para cada etapa de API asociada con las claves de API existentes. En el plan de uso predeterminado, no se establece inicialmente ningún límite de cuota o limitación y las asociaciones entre las claves de la API y las etapas de la API se copian en los planes de uso. La API se comportará igual que antes. Sin embargo, debe utilizar la propiedad `apiStages` de `UsagePlan` para asociar los valores de la etapa de la API especificados (`apiId` y `stage`) con las claves de la API incluidas (mediante `UsagePlanKey`), en lugar de utilizar la propiedad `stageKeys` de ApiKey.

Para comprobar si ya se ha migrado a planes de uso predeterminados, utilice el comando `get-account` de la CLI. En la salida del comando, la lista `features` incluye una entrada de `"UsagePlans"` cuándo los planes de uso están habilitados.

También puede migrar sus API a planes de uso predeterminados utilizando la AWS CLI tal y como se indica a continuación:

Para migrar a planes de uso predeterminados mediante la AWS CLI

1. Llame a este comando de la CLI: `update-account`.
2. Para el parámetro `cli-input-json`, utilice el siguiente JSON:

```
[
    {
        "op": "add",
```

```
        "path": "/features",
        "value": "UsagePlans"
    }
]
```

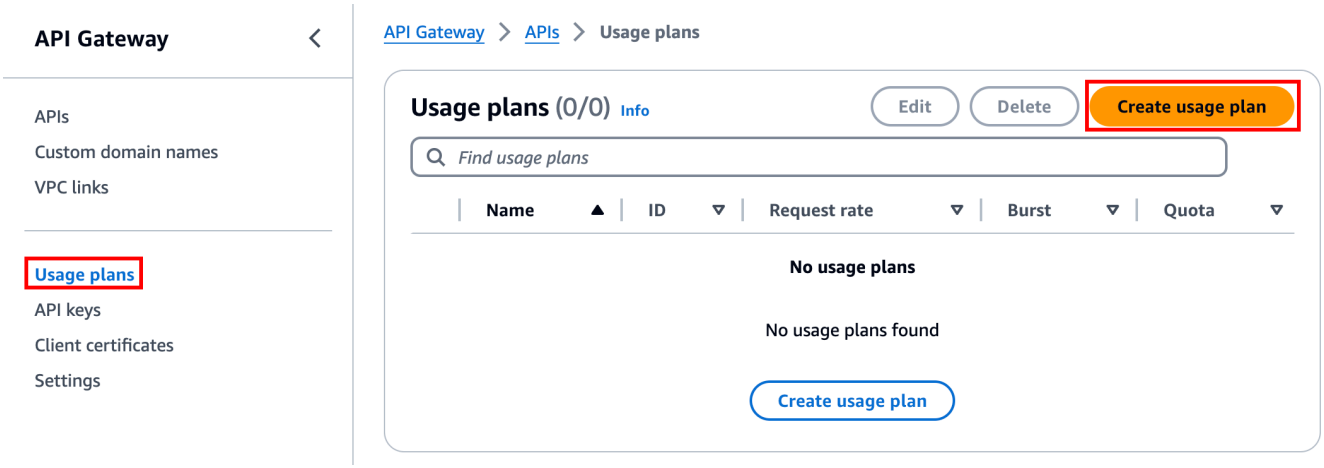
Crear un plan de uso

El siguiente procedimiento describe cómo crear un plan de uso.

Consola de administración de AWS

Para crear un plan de uso

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal de API Gateway, elija Planes de uso y, a continuación, elija Crear plan de uso.



3. En Nombre, ingrese un nombre.
4. (Opcional) En Description (Descripción), introduzca una descripción.
5. De forma predeterminada, los planes de uso habilitan la limitación. Ingrese una tasa y una ráfaga para el plan de uso. Elija Limitación para desactivar la limitación.
6. De forma predeterminada, los planes de uso habilitan una cuota durante un periodo de tiempo. En Solicitudes, ingrese el número total de solicitudes que un usuario puede realizar en el periodo de tiempo del plan de uso. Elija Cuota para desactivar la cuota.
7. Elija Crear plan de uso.

AWS CLI

El siguiente comando create-usage-plan crea un plan de uso que se reinicia a principios de mes:

```
aws apigateway create-usage-plan \
    --name "New Usage Plan" \
    --description "A new usage plan" \
    --throttle burstLimit=10,rateLimit=5 \
    --quota limit=500,offset=0,period=MONTH
```

REST API

Llame a `usageplan:create` para crear un plan de uso.

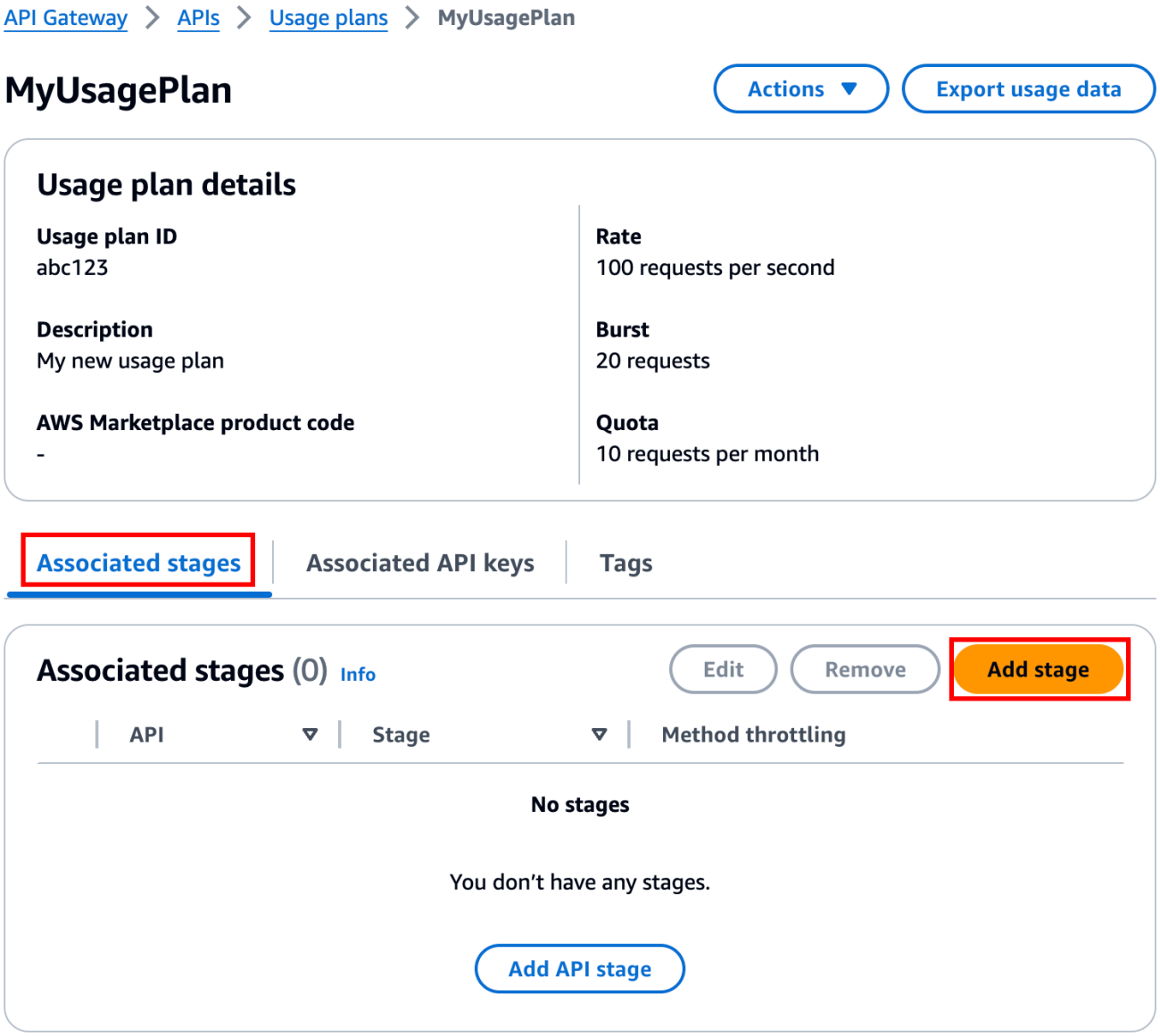
Adición de una etapa a un plan de uso

El siguiente procedimiento describe cómo agregar una etapa a un plan de uso.

Consola de administración de AWS

Adición de una etapa a un plan de uso

1. Seleccione el plan de uso.
2. En la pestaña Etapas asociadas, elija Agregar etapa.



3. En API, seleccione una API.
4. En Etapa, seleccione una etapa.
5. (Opcional) Para activar la regulación en el nivel de método, haga lo siguiente:
   a. Elija Limitación en el nivel de método y, a continuación, elija Agregar método.
   b. En Recurso, seleccione un recurso de la API.
   c. En Método, seleccione un método de la API.
   d. Ingrese una tasa y una ráfaga para el plan de uso.
6. Elija Agregar al plan de uso.

AWS CLI

El siguiente comando update-usage-plan agrega la etapa `Prod` de una API a un plan de uso:

```
aws apigateway update-usage-plan \
    --usage-plan-id abc123 \
    --patch-operations op="add",path="/apiStages",value="a1b1c2:Prod"
```

REST API

Llame a `usageplan:update` para actualizar un plan de uso.

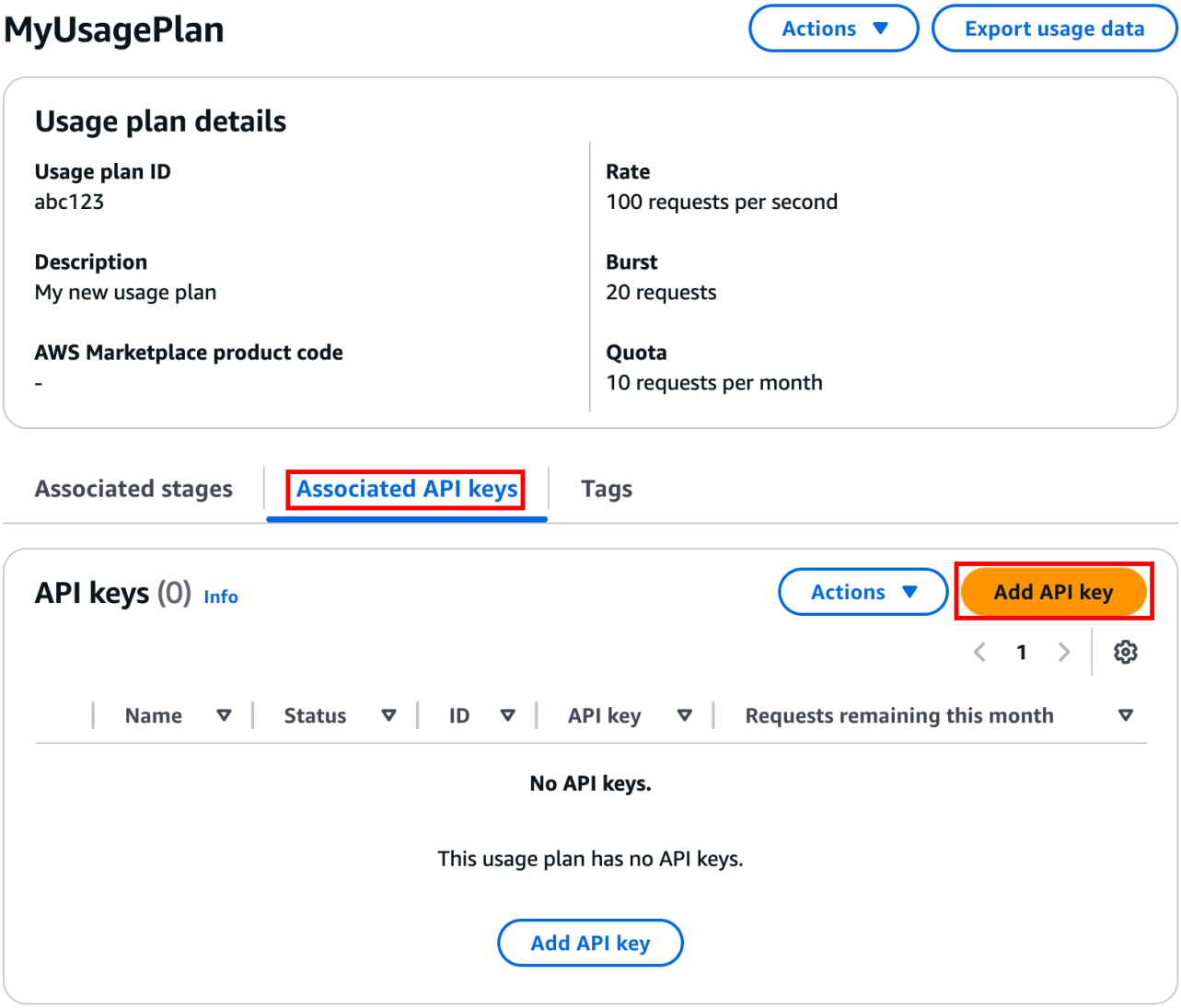
Adición de una clave API a un plan de uso

El procedimiento siguiente muestra cómo agregar una clave de API a un plan de uso.

Consola de administración de AWS

Adición de una clave a un plan de uso

1. En la pestaña Claves de API asociadas, elija Agregar clave de API.



2. a. Para asociar una clave existente al plan de uso, seleccione Agregar clave existente y, a continuación, seleccione la clave existente en el menú desplegable.

   b. Para crear una clave de API nueva, seleccione Crear y agregar una clave nueva y, a continuación, cree una clave nueva. Para obtener más información sobre cómo crear una clave nueva, consulte Crear una clave de API.

3. Elija Agregar clave de la API.

AWS CLI

El siguiente comando create-usage-plan-key asocia una clave de API existente a un plan de uso:

```
aws apigateway create-usage-plan-key \
```

```
    --usage-plan-id a1b2c3 \
    --key-type "API_KEY" \
    --key-id aaa111bbb
```

REST API

Llame a `usageplankey:create` para asociar una clave de API existente a un plan de uso.

También puede asociar directamente las claves de API a un plan de uso cuando las importe. Llame a `apikey:import` para agregar una o varias claves de API directamente al plan de uso especificado. La carga de solicitudes debe incluir los valores de las claves de API, el identificador del plan de uso asociado, los indicadores booleanos que indican que las claves están habilitadas para el plan de uso y, posiblemente, los nombres y descripciones de las claves de API.

En el siguiente ejemplo de la solicitud `apikey:import`, se agregarán tres claves de API (identificadas mediante `key`, `name`y `description`) a un plan de uso (identificado mediante `usageplanIds`):

```
POST /apikeys?mode=import&format=csv&failonwarnings=fase HTTP/1.1
Host: apigateway.us-east-1.amazonaws.com
Content-Type: text/csv
Authorization: ...

key,name, description, enabled, usageplanIds
abcdef1234ghijklmnop8901234567, importedKey_1, firstone,  tRuE, n371pt
abcdef1234ghijklmnop0123456789, importedKey_2, secondone, TRUE, n371pt
abcdef1234ghijklmnop9012345678, importedKey_3,          , true, n371pt
```

Como resultado, se crearán tres recursos `UsagePlanKey` y se agregarán a `UsagePlan`.

También puede añadir claves de API a varios planes de uso de esta manera. Para ello, cambie cada valor de la columna `usageplanIds` por una cadena separada por comas que contenga los identificadores del plan de uso seleccionado entre comillas (`"n371pt,m282qs"` o `'n371pt,m282qs'`).

> **Note**
>
> Una clave de API se puede asociar a más de un plan de uso. Un plan de uso se puede asociar a más de una etapa. Sin embargo, una clave de API determinada solo se puede asociar a un plan de uso para cada etapa de la API.

Mantenimiento de un plan de uso para las API de REST en API Gateway

El mantenimiento de un plan de uso implica monitorear las cuotas usadas y restantes durante un período de tiempo determinado, si es necesario, y ampliar las cuotas restantes en una cantidad específica. Los siguientes procedimientos describen cómo monitorear las cuotas.

Consola de administración de AWS

Para monitorear las cuotas usadas y restantes

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal de API Gateway, elija Planes de uso.
3. Seleccione un plan de uso.
4. Elija la pestaña Claves de API asociadas para ver el número de solicitudes pendientes de cada clave durante el periodo de tiempo.
5. (Opcional) Elija Exportar datos de uso y, a continuación, elija una fecha de inicio y una fecha de finalización. A continuación, elija JSON o CSV como formato de datos exportados y, a continuación, elija Exportar.

   El siguiente ejemplo muestra un archivo exportado.

```
{
    "px1KW6...qBazOJH": [
      [
        0,
        5000
      ],
      [
        0,
        5000
      ],
      [
```

```
        0,
        10
      ]
    ]
  }
```

Los datos de uso del ejemplo representan los datos de uso diario de un cliente de API identificado por la clave de API (`px1KW6...qBaz0JH`), entre el 1 de agosto de 2016 y el 3 de agosto de 2016. Cada dato de uso diario muestra las cuotas usadas y restantes. En este ejemplo, el suscriptor aún no ha utilizado ninguna de las cuotas asignadas y el propietario o administrador de la API redujo la cuota restante de 5000 a 10 durante el tercer día.

Los siguientes procedimientos describen cómo modificar las cuotas.

Para ampliar las cuotas restantes

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal de API Gateway, elija Planes de uso.
3. Seleccione un plan de uso.
4. Elija la pestaña Claves de API asociadas para ver el número de solicitudes pendientes de cada clave durante el periodo de tiempo.
5. Seleccione una clave de API y, a continuación, elija Conceder extensión de uso.
6. Escriba un número para la cuota de las solicitudes restantes. Puede aumentar las solicitudes de cambio de nombre o reducir las solicitudes restantes durante el periodo de tiempo del plan de uso.
7. Elija Actualizar cuota.

AWS CLI

Los siguientes ejemplos update-usage-plan permiten agregar, quitar o modificar la configuración de limitación de nivel de método en un plan de uso.

> **Note**
>
> Asegúrese de cambiar `us-east-1` al valor de región adecuado para la API.

Para agregar o sustituir un límite de frecuencia a fin de limitar de forma controlada un recurso y un método individuales:

```
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"
```

Para agregar o sustituir un límite de ráfaga a fin de limitar de forma controlada un recurso y un método individuales:

```
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"
```

Para quitar la configuración de limitación controlada de nivel de método para un recurso y un método individuales:

```
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=""
```

Para quitar toda la configuración de limitación controlada de nivel de método para una API:

```
aws apigateway --region us-east-1 update-usage-plan --usage-plan-id planId --patch-
operations op="remove",path="/apiStages/apiId:stage/throttle ",value=""
```

A continuación se muestra un ejemplo con la API de ejemplo PetStore:

```
aws apigateway --region us-east-1 update-usage-plan --usage-plan-id planId --patch-
operations op="replace",path="/apiStages/apiId:stage/throttle",value='"{\"/pets/GET
\":{\"rateLimit\":1.0,\"burstLimit\":1},\"//GET\":{\"rateLimit\":1.0,\"burstLimit
\":1}}"'
```

REST API

Llame a `usageplan:update` para mantener un plan de uso.

Creación y configuración de claves de API y planes de uso con CloudFormation

Puede utilizar CloudFormation para solicitar claves de API en los métodos de la API y crear un plan de uso para una API. En la plantilla de CloudFormation de ejemplo se realiza lo siguiente:

- Crea una API de API Gateway con los métodos `GET` y `POST`.
- Requiere una clave de API para los métodos `GET` y `POST`. Esta API recibe claves del encabezado `X-API-KEY` de cada solicitud entrante.
- Crea una clave de API.
- Crea un plan de uso para especificar una limitación de 1000 solicitudes al mes, una limitación de velocidad controlada de 100 solicitudes por segundo y una limitación de ráfagas controlada de 200 solicitudes por segundo.
- Especifica una limitación controlada de velocidad en el nivel de método de 50 solicitudes por segundo y una limitación controlada de ráfaga en el nivel de método de 100 solicitudes por segundo para el método `GET`.
- Asocia la etapa de API y la clave de API con el plan de uso.

```
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}'
```

Configuración de un método para usar claves de API con una definición de OpenAPI

Puede usar una definición de OpenAPI para requerir claves de API en un método.

Para cada método, cree un objeto de requisito de seguridad que requiera una clave de API para invocar ese método. A continuación, defina `api_key` en la definición de seguridad. Después de crear la API, agregue la etapa de API nueva al plan de uso.

En el siguiente ejemplo, se crea una API y se requiere una clave de API para los métodos `POST` y `GET`:

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"
      }
    }
  }
}
```

Prueba de planes de uso para API de REST en API Gateway

A modo de ejemplo, vamos a utilizar la API PetStore, que se creó en Tutorial: Crear una API de REST importando un ejemplo. Supongamos que la API está configurada para utilizar una clave de API de `Hiorr45VR...c4GJc`. Los siguientes pasos describen cómo probar un plan de uso.

Para probar su plan de uso

- Realice una solicitud `GET` en el recurso Pets (`/pets`), con los parámetros de consulta `?type=...&page=...` de la API (por ejemplo, `xbvxlpijch`) en un plan de uso:

```
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**
>
> Debe enviar esta solicitud al componente `execute-api` de la API Gateway y proporcionar la clave de API necesaria (por ejemplo, `Hiorr45VR...c4GJc`) en el encabezado `x-api-key` correspondiente.

La respuesta, si se ejecuta correctamente, devuelve un código de estado `200 OK` y una carga que contiene los resultados solicitados del backend. Si olvidó establecer el encabezado `x-api-key` o lo estableció con una clave incorrecta, obtendrá una respuesta `403 Forbidden`. Sin embargo, si no configuró el método para que solicite una clave de API, probablemente obtendrá una respuesta `200 OK` tanto si estableció el encabezado `x-api-key` correctamente como si no, y las limitaciones de solicitudes y los límites de cuota del plan de uso se omitirán.

Ocasionalmente, cuando se produce un error interno que impide a API Gateway imponer limitaciones controladas en el plan de uso o límites de cuota en la solicitud, API Gateway proporciona la solicitud sin aplicar los límites ni las cuotas especificados en el plan de uso. Sin embargo, registra un mensaje de error `Usage Plan check failed due to an internal error` en CloudWatch. Puede obviar este tipo de errores ocasionales.

Llamada a un método con una clave de API

En función del tipo de origen de clave de API que elija, utilice uno de los siguientes procedimientos para usar claves de API proporcionadas por un encabezado o claves de API proporcionadas por un autorizador en la invocación de método:

Para utilizar claves de API proporcionadas por un encabezado:

1. Cree una API con los métodos de la API que desee y, a continuación, implemente la API en una etapa.
2. Cree un nuevo plan de uso o elija uno existente. Añada la etapa de API implementada al plan de uso. Asocie una clave de API al plan de uso o elija una clave de API existente en el plan. Anote el valor de la clave de API elegida.
3. Configure los métodos de la API para que exijan una clave de API.
4. Vuelva a implementar la API en la misma etapa. Si implementa la API en una nueva etapa, asegúrese de actualizar el plan de uso para asociar la nueva etapa de API.
5. Llame a la API utilizando la clave de API. El siguiente comando curl de ejemplo invoca el método `GET` en el recurso `getUsers` de la etapa `prod` de una API utilizando una clave de API.

```
curl -H "X-API-Key: abcd1234" 'https://b123abcde4.execute-api.us-
west-2.amazonaws.com/prod/getUsers'
```

El cliente puede llamar ahora a los métodos de la API proporcionando al encabezado `x-api-key` la clave de API elegida como valor de encabezado. Una llamada tendría un aspecto similar al siguiente:

Para utilizar claves de API proporcionadas por un autorizador:

1. Cree una API con los métodos de la API que desee y, a continuación, implemente la API en una etapa.

2. Cree un nuevo plan de uso o elija uno existente. Añada la etapa de API implementada al plan de uso. Asocie una clave de API al plan de uso o elija una clave de API existente en el plan. Anote el valor de la clave de API elegida.
3. Cree un autorizador de Lambda basado en token Incluya `usageIdentifierKey:{api-key}` como propiedad de nivel de raíz de la respuesta de autorización. Para obtener instrucciones sobre cómo crear un autorizador basado en token, consulte the section called “Ejemplo de función de Lambda con un autorizador TOKEN”.
4. Configure los métodos de la API para que exijan una clave de API y habilite el autorizador de Lambda en los métodos.
5. Vuelva a implementar la API en la misma etapa. Si implementa la API en una nueva etapa, asegúrese de actualizar el plan de uso para asociar la nueva etapa de API.

El cliente puede llamar ahora a los métodos requeridos por la clave de API sin proporcionar explícitamente una clave de API. La clave de API proporcionada por un autorizador se usa de forma automática.

Documentación para las API de REST en API Gateway

Para ayudar a los clientes a comprender y utilizar la API, debe documentar la API. Para ayudar a documentar su API, API Gateway le permite agregar y actualizar el contenido de la ayuda de las distintas entidades de API como parte integral de su proceso de desarrollo de la API. API Gateway almacena el contenido de origen y le permite archivar diferentes versiones de la documentación. Puede asociar una versión de la documentación a una etapa de la API, exportar una instantánea de documentación específica de la etapa a un archivo de OpenAPI externo y distribuir el archivo como una publicación de la documentación.

Para documentar la API, puede llamar a la API de REST de API Gateway, utilizar uno de los AWS SDK, la AWS CLI para API Gateway o usar la consola de API Gateway. Además, puede importar o exportar las piezas de la documentación definidas en un archivo de OpenAPI externo.

Para compartir la documentación de la API con los desarrolladores, puede usar un portal para desarrolladores. Para ver un ejemplo, consulte Integrating ReadMe with API Gateway to Keep Your Developer Hub Up to Date o How to Streamline API Development on Amazon API Gateway Using SmartBear’s SwaggerHub en el blog de Red de socios (APN) de AWS.

Temas

- Representación de la documentación de la API en API Gateway

- Proceso de documentación de una API mediante la consola API Gateway
- Publicación de documentación de la API mediante la consola de API Gateway
- Proceso de documentación de una API mediante la API de REST de API Gateway
- Publicación de documentación de la API mediante la API de REST de API Gateway
- Importar la documentación de API
- Controlar el acceso a la documentación de la API en API Gateway

Representación de la documentación de la API en API Gateway

La documentación de la API de API Gateway consta de piezas de documentación individuales asociadas a entidades de API específicas que incluyen la API, el recurso, el método, la solicitud, la respuesta, los parámetros de mensaje (como ruta, consulta, encabezado), así como autorizadores y modelos.

En API Gateway, una pieza de documentación se representa mediante un recurso DocumentationPart. La documentación de la API en su conjunto se representa mediante la colección DocumentationParts.

Documentar una API implica crear instancias de `DocumentationPart`, añadirlas a la colección `DocumentationParts` y mantener versiones de las piezas de la documentación a medida que evolucione la API.

Temas

- Piezas de la documentación
- Versiones de la documentación

Piezas de la documentación

Un recurso DocumentationPart es un objeto JSON que almacena el contenido de la documentación aplicable a una entidad de API determinada. Esto incluye cualquier contenido UTF-8 y todos los idiomas de localización principales de la documentación. Su campo `properties` incluye el contenido de la documentación como un mapa de pares de clave-valor. Su propiedad `location` identifica la entidad de API asociada.

La forma de un mapa de contenido lo determina usted, el desarrollador de la API. El valor del par de clave-valor puede ser una cadena, un número, un valor booleano, un objeto o una matriz. La forma del objeto `location` depende del tipo de entidad de destino.

El recurso `DocumentationPart` admite la herencia de contenido: el contenido de documentación de una entidad de API se aplica a los elementos secundarios de dicha entidad. Para obtener más información sobre la definición de entidades secundarias y la herencia de contenido, consulte Heredar contenido de una entidad de API de especificación más general.

Ubicación de una pieza de documentación

La propiedad location de una instancia DocumentationPart identifica una entidad de API a la que se aplica el contenido asociado. La entidad de la API puede ser un recurso de la API de REST de API Gateway, como RestApi, Resource, Method, MethodResponse, Authorizer o Model. La entidad también puede ser un parámetro de mensaje, como un parámetro de ruta URL, un parámetro de cadena de consulta, un parámetro de encabezado de solicitud o respuesta, una solicitud o cuerpo de respuesta o un código de estado de respuesta.

Para especificar una entidad de API, establezca el atributo type del objeto `location` en `API`, `AUTHORIZER`, `MODEL`, `RESOURCE`, `METHOD`, `PATH_PARAMETER`, `QUERY_PARAMETER`, `REQUEST_HEADER`, `REQUEST_BODY`, `RESPONSE`, `RESPONSE_HEADER` o `RESPONSE_BODY`.

En función del `type` de una entidad de API, puede especificar otros atributos `location`, incluidos method, name, path y statusCode. No todos estos atributos son válidos para una determinada entidad de API. Por ejemplo, `type`, `path`, `name` y `statusCode` son atributos válidos de la entidad `RESPONSE`; solo `type` y `path` son atributos de ubicación válidos de la entidad `RESOURCE`. Es un error incluir un campo no válido en el atributo `location` de un recurso `DocumentationPart` para una determinada entidad de API.

No todos los campos `location` válidos son obligatorios. Por ejemplo, `type` es el campo `location` válido y obligatorio para todas las entidades de API. Sin embargo, `method`, `path` y `statusCode` son válidos, pero no son atributos obligatorios para la entidad `RESPONSE`. Cuando no se especifica explícitamente, un campo `location` válido asume el valor predeterminado. El valor `path` predeterminado es `/`, es decir, el recurso raíz de una API. El valor predeterminado de `method` o `statusCode` es `*`, que indica cualquier método o valores de código de estado, respectivamente.

Contenido de una pieza de documentación

El valor `properties` está codificado como una cadena JSON. El valor `properties` contiene la información que elija para satisfacer sus necesidades de documentación. Por ejemplo, a continuación se muestra un mapa de contenido válido:

```
{
  "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."
}
```

Aunque API Gateway admite cualquier cadena JSON válida como mapa de contenido, los atributos de contenido se tratan atendiendo a dos categorías: los que OpenAPI reconoce y los que no. En el ejemplo anterior, `info`, `description` y `x-custom-info` son atributos reconocidos por OpenAPI como un objeto, propiedad o extensión estándar de OpenAPI. En cambio, `my-info` no es compatible con la especificación de OpenAPI. API Gateway propaga los atributos de contenido compatibles con OpenAPI a las definiciones de entidad de API de las instancias de `DocumentationPart` asociadas. API Gateway no propaga los atributos de contenido no compatibles a las definiciones de entidad de API.

Veamos otro ejemplo; aquí `DocumentationPart` es el destino de una entidad `Resource`:

```
{
    "location" : {
        "type" : "RESOURCE",
        "path": "/pets"
    },
    "properties" : {
        "summary" : "The /pets resource represents a collection of pets in PetStore.",
        "description": "... a child resource under the root...",
    }
}
```

Tanto `type` como `path` son campos válidos para identificar el destino del tipo `RESOURCE`. Para el recurso raíz (`/`), puede omitir el campo `path`.

```
{
    "location" : {
        "type" : "RESOURCE"
    },
    "properties" : {
        "description" : "The root resource with the default path specification."
    }
}
```

Es el mismo que la siguiente instancia de `DocumentationPart`:

```
{
    "location" : {
        "type" : "RESOURCE",
        "path": "/"
    },
    "properties" : {
        "description" : "The root resource with an explicit path specification"
    }
}
```

Heredar contenido de una entidad de API de especificaciones más generales

El valor predeterminado de un campo `location` opcional proporciona una descripción de los patrones de una entidad de API. Con el valor predeterminado en el objeto `location`, puede añadir una descripción general en el mapa `properties` a una instancia de `DocumentationPart` con este tipo de patrón `location`. API Gateway extrae los atributos de documentación de OpenAPI aplicables de `DocumentationPart` de la entidad de API genérica y los inserta en una entidad de API específica con los campos `location` con valores que coinciden con el patrón `location` general o con el valor exacto, a menos que la entidad especificada ya tenga una instancia de `DocumentationPart` asociada. Este comportamiento se denomina también "herencia de contenido de una entidad de API de especificaciones más generales".

La herencia de contenido no se aplica a determinados tipos de entidades de API. Consulte la siguiente tabla para obtener más detalles.

Cuando una entidad de API coincide con más de un patrón de ubicación de `DocumentationPart`, la entidad heredará la pieza de documentación con los campos de ubicación que tengan mayor prioridad y sean más específicos. El orden de prioridad es `path` > `statusCode`. Para la correspondencia con el campo `path`, API Gateway elige la entidad con el valor de ruta más específico. En la siguiente tabla se muestra esto con algunos ejemplos.

Caso	**path**	**state**	**name**	Remar
1	`/pets`	`*`	`id`	La documentación asociada

Caso	`path`	`state`	`name`	Remar
				a este patrón de ubicación la heredarán las entidades que coincidan con el patrón de ubicación.

Caso	**path**	**state**	**name**	Remar
2	/pets	200	id	La documentación asociada a este patrón de ubicación la heredarán las entidades que coincidan con el patrón de ubicación cuando coincidan tanto Caso 1 como Caso 2, ya que Caso 2 es más específic

Caso	`path`	`state`	`name`	Remar
				o que Caso 1.

Caso	**path**	**state**	**name**	Remar
3	/pets/petId	*	id	La documentación asociada a este patrón de ubicación la heredarán las entidades que coincidan con el patrón de ubicación cuando los tres casos coincidan, ya que Caso 3 tiene más prioridad que

Caso	**path**	**state**	**name**	Remar
				Caso 2 y es más específico que Caso 1.

Este es otro ejemplo en el que se compara una instancia de `DocumentationPart` más genérica con una más específica. El siguiente mensaje de error general `"Invalid request error"` se inserta en las definiciones de OpenAPI de las respuestas de error `400`, a menos que se invalide.

```
{
    "location" : {
        "type" : "RESPONSE",
        "statusCode": "400"
    },
    "properties" : {
        "description" : "Invalid request error."
    }"
}
```

Con la siguiente invalidación, las respuestas `400` a cualquier método del recurso `/pets` tienen una descripción de `"Invalid petId specified"` en su lugar.

```
{
    "location" : {
        "type" : "RESPONSE",
        "path": "/pets",
        "statusCode": "400"
    },
    "properties" : "{
        "description" : "Invalid petId specified."
    }"
```

```
}
```

Campos de ubicación válidos de **`DocumentationPart`**

En la siguiente tabla, se muestran los campos válidos y obligatorios, así como los valores predeterminados aplicables, de un recurso DocumentationPart asociado a un tipo de entidades de API determinado.

Entidad de API	Campos de ubicación válidos	Campos de ubicación obligatorios	Valores de campo predeterminados	Contenido heredable
API	`{ "location": { "type": "API" }, ... }`	`type`	N/A	No
Recurso	`{ "location": { "type": "RESOURCE", "path": "resource_path " }, ... }`	`type`	El valor predeterminado de `path` es `/`.	No
Método	`{ "location": { "type": "METHOD", "path": "resource_path ", "method": "http_verb " }, ...`	`type`	Los valores predeterminados de `path` y `method` son `/` y `*`, respectivamente.	Sí, buscando coincidencias de `path` por prefijo y buscando coincidencias del elemento `method` de

Entidad de API	Campos de ubicación válidos	Campos de ubicación obligatorios	Valores de campo predeterminados	Contenido heredable
	`}`			cualquier valor.
Parámetros de consulta	`{ "location": { "type": "QUERY_PARAMETER", "path": "resource_path ", "method": "HTTP_verb ", "name": "query_parameter_name " }, ... }`	`type`	Los valores predeterminados de `path` y `method` son `/` y `*`, respectivamente.	Sí, buscando coincidencias de `path` por prefijo y buscando coincidencias del elemento `method` por los valores exactos.
Cuerpo de la solicitud	`{ "location": { "type": "REQUEST_BODY", "path": "resource_path ", "method": "http_verb " }, ... }`	`type`	Los valores predeterminados de `path` y `method` son `/` y `*`, respectivamente.	Sí, buscando coincidencias de `path` por prefijo y buscando coincidencias del elemento `method` por los valores exactos.

Entidad de API	Campos de ubicación válidos	Campos de ubicación obligatorios	Valores de campo predeterminados	Contenido heredable
Parámetro de encabezado de solicitud	`{ "location": { "type": "REQUEST_HEADER", "path": "resource_path ", "method": "HTTP_verb ", "name": "header_name " }, ... }`	`type`, `name`	Los valores predeterminados de `path` y `method` son `/` y `*`, respectivamente.	Sí, buscando coincidencias de `path` por prefijo y buscando coincidencias del elemento `method` por los valores exactos.
Parámetros de ruta de solicitud	`{ "location": { "type": "PATH_PARAMETER", "path": "resource/{path_parameter_name }", "method": "HTTP_verb ", "name": "path_parameter_name " }, ... }`	`type`, `name`	Los valores predeterminados de `path` y `method` son `/` y `*`, respectivamente.	Sí, buscando coincidencias de `path` por prefijo y buscando coincidencias del elemento `method` por los valores exactos.

Entidad de API	Campos de ubicación válidos	Campos de ubicación obligatorios	Valores de campo predeterminados	Contenido heredable
Respuesta	`{ "location": { "type": "RESPONSE", "path": "resource_path", "method": "http_verb", "statusCode": "status_code" }, ... }`	`type`	Los valores predeterminados de `path`, `method` y `statusCode` son `/`, * y *, respectivamente.	Sí, buscando coincidencias de `path` por prefijo y buscando coincidencias de los elementos `method` y `statusCode` por los valores exactos.
Encabezado de respuesta	`{ "location": { "type": "RESPONSE_HEADER", "path": "resource_path", "method": "http_verb", "statusCode": "status_code", "name": "header_name" }, ... }`	`type`, `name`	Los valores predeterminados de `path`, `method` y `statusCode` son `/`, * y *, respectivamente.	Sí, buscando coincidencias de `path` por prefijo y buscando coincidencias de los elementos `method` y `statusCode` por los valores exactos.

Entidad de API	Campos de ubicación válidos	Campos de ubicación obligatorios	Valores de campo predeterminados	Contenido heredable
Cuerpo de respuesta	`{ "location": { "type": "RESPONSE_BODY", "path": "resource_path ", "method": "http_verb ", "statusCode": "status_code " }, ... }`	`type`	Los valores predeterminados de `path`, `method` y `statusCode` son `/`, * y *, respectivamente.	Sí, buscando coincidencias de `path` por prefijo y buscando coincidencias de los elementos `method` y `statusCode` por los valores exactos.
Autorizador	`{ "location": { "type": "AUTHORIZER", "name": "authorizer_name " }, ... }`	`type`	N/A	No
Modelo	`{ "location": { "type": "MODEL", "name": "model_name " }, ... }`	`type`	N/A	No

Versiones de la documentación

Una versión de la documentación es una instantánea de la colección DocumentationParts de una API, que está etiquetada con un identificador de versión. Publicar la documentación de una API implica crear una versión de documentación, asociarla a una etapa de la API y exportar esa versión específica de la etapa de la documentación de la API a un archivo de OpenAPI externo. En API Gateway, una instantánea de documentación se representa como un recurso DocumentationVersion.

Cuando se actualiza una API, se crean nuevas versiones de la API. En API Gateway, todas las versiones de la documentación se mantienen mediante la colección DocumentationVersions.

Proceso de documentación de una API mediante la consola API Gateway

En esta sección se describe cómo crear y actualizar piezas de documentación de una API mediante la consola de API Gateway.

Un requisito previo para crear y editar la documentación de una API es que ya debe haber creado la API. En esta sección, usaremos la API PetStore como ejemplo. Para crear una API mediante la consola de API Gateway, siga las instrucciones de Tutorial: Crear una API de REST importando un ejemplo.

Temas

- Documentar la entidad API
- Documentar una entidad RESOURCE
- Documentar una entidad METHOD
- Documentar una entidad QUERY_PARAMETER
- Documentar una entidad PATH_PARAMETER
- Documentar una entidad REQUEST_HEADER
- Documentar una entidad REQUEST_BODY
- Documentar una entidad RESPONSE
- Documentar una entidad RESPONSE_HEADER
- Documentar una entidad RESPONSE_BODY
- Documentar una entidad MODEL
- Documentar una entidad AUTHORIZER

Documentar la entidad **API**

Para agregar una nueva pieza de documentación para la entidad `API`, haga lo siguiente:

1. En el panel de navegación principal, elija Documentación y, a continuación, elija Crear pieza de documentación.
2. En Tipo de documentación, seleccione API.

   Si no se creó una pieza de documentación para la `API`, verá el editor del mapa `properties` de la pieza de documentación. Especifique el siguiente mapa `properties` en el editor de texto.

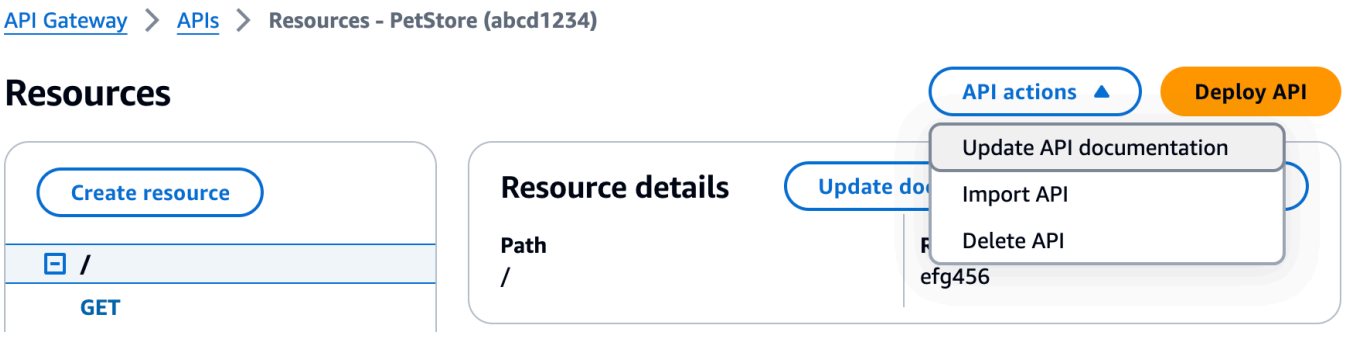
   ```
   {
     "info": {
       "description": "Your first API Gateway API.",
       "contact": {
           "name": "John Doe",
           "email": "john.doe@api.com"
       }
     }
   }
   ```

   > **Note**
   >
   > No es necesario codificar el mapa `properties` en una cadena JSON. La consola de API Gateway representa el objeto JSON en una cadena automáticamente.

3. Elija Crear pieza de documentación.

Para agregar una nueva pieza de documentación para la entidad `API` en el panel Recursos, haga lo siguiente:

1. En el panel de navegación principal, elija Recursos.
2. Elija el menú Acciones API y, a continuación, elija Actualizar la documentación de la API.



Para editar una pieza existente de la documentación, haga lo siguiente:

1. En el panel Documentación, elija la pestaña Recursos y métodos.
2. Seleccione el nombre de la API y, a continuación, en la tarjeta de API, elija Editar.

Documentar una entidad **`RESOURCE`**

Para agregar una nueva pieza de documentación para una entidad `RESOURCE`, haga lo siguiente:

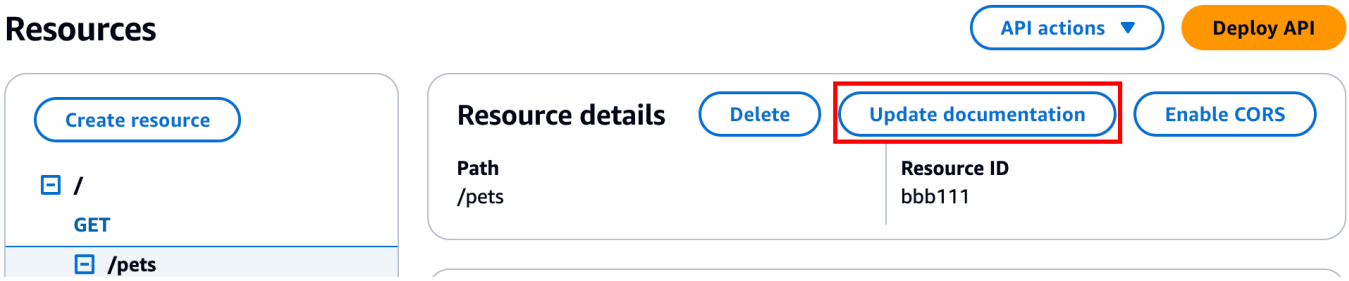
1. En el panel de navegación principal, elija Documentación y, a continuación, elija Crear pieza de documentación.
2. En Tipo de documentación, seleccione API.
3. En Ruta, ingrese una ruta.
4. Ingrese una descripción en el editor de texto, por ejemplo:

```
{
    "description": "The PetStore's root resource."
}
```

5. Elija Crear pieza de documentación. Puede crear documentación para un recurso que no figure en la lista.
6. Si es necesario, repita estos pasos para agregar o editar otra pieza de documentación.

Para agregar una nueva pieza de documentación para una entidad `RESOURCE` en el panel Recursos, haga lo siguiente:

1. En el panel de navegación principal, elija Recursos.
2. Elija el recurso y, a continuación, elija Actualizar documentación.



Para editar una pieza existente de la documentación, haga lo siguiente:

1. En el panel Documentación, elija la pestaña Recursos y métodos.
2. Seleccione el recurso que contiene la pieza de documentación y, a continuación, elija Editar.

Documentar una entidad **`METHOD`**

Para agregar una nueva pieza de documentación para una entidad `METHOD`, haga lo siguiente:

1. En el panel de navegación principal, elija Documentación y, a continuación, elija Crear pieza de documentación.
2. En Tipo de documentación, seleccione Método.
3. En Ruta, ingrese una ruta.
4. En Método, seleccione un verbo HTTP.
5. Ingrese una descripción en el editor de texto, por ejemplo:

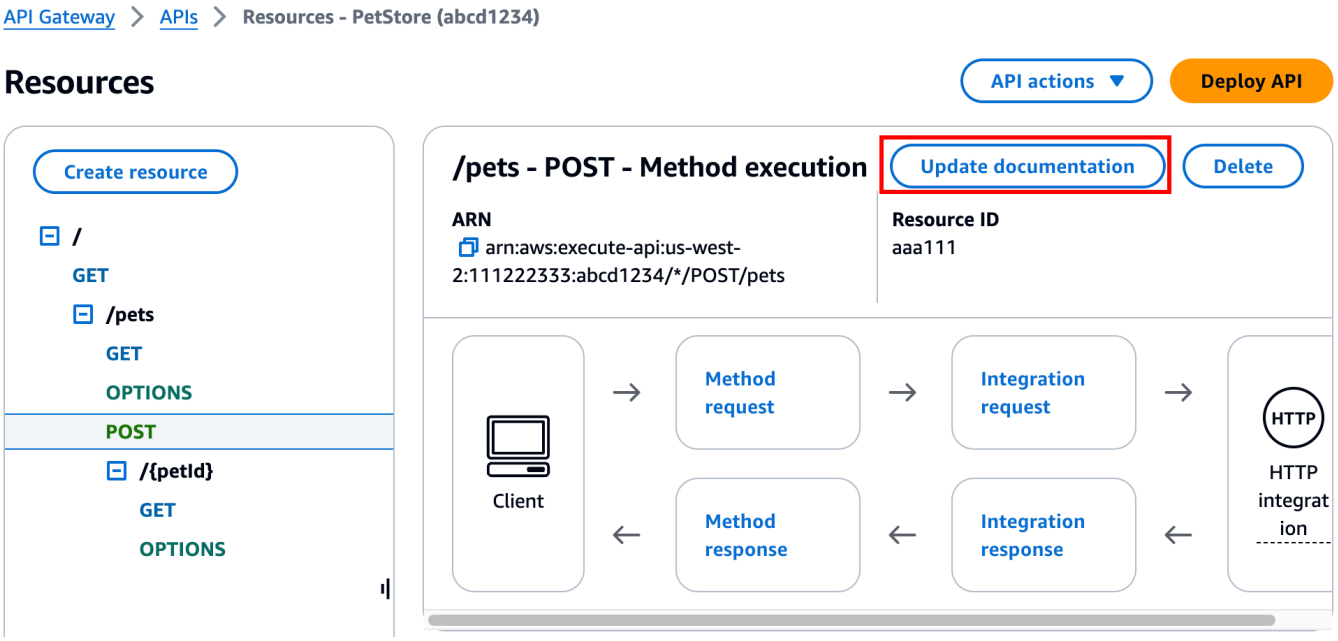
```
{
  "tags" : [ "pets" ],
  "summary" : "List all pets"
}
```

6. Elija Crear pieza de documentación. Puede crear documentación para un método que no figure en la lista.
7. Si es necesario, repita estos pasos para agregar o editar otra pieza de documentación.

Para agregar una nueva pieza de documentación para una entidad `METHOD` en el panel Recursos, haga lo siguiente:

1. En el panel de navegación principal, elija Recursos.

2. Elija el método y, a continuación, elija Actualizar documentación.



Para editar una pieza existente de la documentación, haga lo siguiente:

1. En el panel Documentación, elija la pestaña Recursos y métodos.
2. Puede seleccionar el método o seleccionar el recurso que contiene el método y, a continuación, utilizar la barra de búsqueda para buscar y seleccionar la pieza de documentación.
3. Elija Edit (Edición de).

Documentar una entidad `QUERY_PARAMETER`

Para agregar una nueva pieza de documentación para una entidad `QUERY_PARAMETER`, haga lo siguiente:

1. En el panel de navegación principal, elija Documentación y, a continuación, elija Crear pieza de documentación.
2. En Tipo de documentación, seleccione Parámetro de consulta.
3. En Ruta, ingrese una ruta.
4. En Método, seleccione un verbo HTTP.
5. En Nombre, ingrese un nombre.
6. Ingrese una descripción en el editor de texto.

7. Elija Crear pieza de documentación. Puede crear documentación para un parámetro de consulta que no figure en la lista.
8. Si es necesario, repita estos pasos para agregar o editar otra pieza de documentación.

Para editar una pieza existente de la documentación, haga lo siguiente:

1. En el panel Documentación, elija la pestaña Recursos y métodos.
2. Puede seleccionar el parámetro de consulta o seleccionar el recurso que contiene el parámetro de consulta y, a continuación, utilizar la barra de búsqueda para buscar y seleccionar la pieza de documentación.
3. Elija Edit (Edición de).

Documentar una entidad **`PATH_PARAMETER`**

Para agregar una nueva pieza de documentación para una entidad `PATH_PARAMETER`, haga lo siguiente:

1. En el panel de navegación principal, elija Documentación y, a continuación, elija Crear pieza de documentación.
2. En Tipo de documentación, seleccione el Parámetro de ruta.
3. En Ruta, ingrese una ruta.
4. En Método, seleccione un verbo HTTP.
5. En Nombre, ingrese un nombre.
6. Ingrese una descripción en el editor de texto.
7. Elija Crear pieza de documentación. Puede crear documentación para un parámetro de ruta que no figure en la lista.
8. Si es necesario, repita estos pasos para agregar o editar otra pieza de documentación.

Para editar una pieza existente de la documentación, haga lo siguiente:

1. En el panel Documentación, elija la pestaña Recursos y métodos.
2. Puede seleccionar el parámetro de ruta o seleccionar el recurso que contiene el parámetro de ruta y, a continuación, utilizar la barra de búsqueda para buscar y seleccionar la pieza de documentación.
3. Elija Edit (Edición de).

Documentar una entidad **REQUEST_HEADER**

Para agregar una nueva pieza de documentación para una entidad REQUEST_HEADER, haga lo siguiente:

1. En el panel de navegación principal, elija Documentación y, a continuación, elija Crear pieza de documentación.
2. En Tipo de documentación, seleccione Encabezado de solicitud.
3. En Ruta, ingrese una ruta para el encabezado de la solicitud.
4. En Método, seleccione un verbo HTTP.
5. En Nombre, ingrese un nombre.
6. Ingrese una descripción en el editor de texto.
7. Elija Crear pieza de documentación. Puede crear documentación para un encabezado de solicitud que no figure en la lista.
8. Si es necesario, repita estos pasos para agregar o editar otra pieza de documentación.

Para editar una pieza existente de la documentación, haga lo siguiente:

1. En el panel Documentación, elija la pestaña Recursos y métodos.
2. Puede seleccionar el encabezado de solicitud o seleccionar el recurso que contiene el encabezado de solicitud y, a continuación, utilizar la barra de búsqueda para buscar y seleccionar la pieza de documentación.
3. Elija Edit (Edición de).

Documentar una entidad **REQUEST_BODY**

Para agregar una nueva pieza de documentación para una entidad REQUEST_BODY, haga lo siguiente:

1. En el panel de navegación principal, elija Documentación y, a continuación, elija Crear pieza de documentación.
2. En Tipo de documentación, seleccione Cuerpo de la solicitud.
3. En Ruta, ingrese una ruta para el cuerpo de la solicitud.
4. En Método, seleccione un verbo HTTP.

5. Ingrese una descripción en el editor de texto.
6. Elija Crear pieza de documentación. Puede crear documentación para un cuerpo de solicitud que no figure en la lista.
7. Si es necesario, repita estos pasos para agregar o editar otra pieza de documentación.

Para editar una pieza existente de la documentación, haga lo siguiente:

1. En el panel Documentación, elija la pestaña Recursos y métodos.
2. Puede seleccionar el cuerpo de la solicitud o seleccionar el recurso que contiene el cuerpo de la solicitud y, a continuación, utilizar la barra de búsqueda para buscar y seleccionar la pieza de documentación.
3. Elija Edit (Edición de).

Documentar una entidad **`RESPONSE`**

Para agregar una nueva pieza de documentación para una entidad `RESPONSE`, haga lo siguiente:

1. En el panel de navegación principal, elija Documentación y, a continuación, elija Crear pieza de documentación.
2. En Tipo de documentación, seleccione Respuesta (código de estado).
3. En Ruta, ingrese una ruta para la respuesta.
4. En Método, seleccione un verbo HTTP.
5. En Código de estado, ingrese un código de estado HTTP.
6. Ingrese una descripción en el editor de texto.
7. Elija Crear pieza de documentación. Puede crear documentación para un código de estado de respuesta que no figure en la lista.
8. Si es necesario, repita estos pasos para agregar o editar otra pieza de documentación.

Para editar una pieza existente de la documentación, haga lo siguiente:

1. En el panel Documentación, elija la pestaña Recursos y métodos.
2. Puede seleccionar el código de estado de la respuesta o seleccionar el recurso que contiene el código de estado de la respuesta y, a continuación, utilizar la barra de búsqueda para buscar y seleccionar la pieza de documentación.

3. Elija Edit (Edición de).

Documentar una entidad **RESPONSE_HEADER**

Para agregar una nueva pieza de documentación para una entidad RESPONSE_HEADER, haga lo siguiente:

1. En el panel de navegación principal, elija Documentación y, a continuación, elija Crear pieza de documentación.
2. En Tipo de documentación, seleccione Encabezado de respuesta.
3. En Ruta, ingrese una ruta para el encabezado de la respuesta.
4. En Método, seleccione un verbo HTTP.
5. En Código de estado, ingrese un código de estado HTTP.
6. Ingrese una descripción en el editor de texto.
7. Elija Crear pieza de documentación. Puede crear documentación para un encabezado de respuesta que no figure en la lista.
8. Si es necesario, repita estos pasos para agregar o editar otra pieza de documentación.

Para editar una pieza existente de la documentación, haga lo siguiente:

1. En el panel Documentación, elija la pestaña Recursos y métodos.
2. Puede seleccionar el encabezado de la respuesta o seleccionar el recurso que contiene el encabezado de la respuesta y, a continuación, utilizar la barra de búsqueda para buscar y seleccionar la pieza de documentación.
3. Elija Edit (Edición de).

Documentar una entidad **RESPONSE_BODY**

Para agregar una nueva pieza de documentación para una entidad RESPONSE_BODY, haga lo siguiente:

1. En el panel de navegación principal, elija Documentación y, a continuación, elija Crear pieza de documentación.
2. En Tipo de documentación, seleccione Cuerpo de respuesta.
3. En Ruta, ingrese una ruta para el cuerpo de la respuesta.

4. En Método, seleccione un verbo HTTP.
5. En Código de estado, ingrese un código de estado HTTP.
6. Ingrese una descripción en el editor de texto.
7. Elija Crear pieza de documentación. Puede crear documentación para un cuerpo de respuesta que no figure en la lista.
8. Si es necesario, repita estos pasos para agregar o editar otra pieza de documentación.

Para editar una pieza existente de la documentación, haga lo siguiente:

1. En el panel Documentación, elija la pestaña Recursos y métodos.
2. Puede seleccionar el cuerpo de la respuesta o seleccionar el recurso que contiene el cuerpo de la respuesta y, a continuación, utilizar la barra de búsqueda para buscar y seleccionar la pieza de documentación.
3. Elija Edit (Edición de).

Documentar una entidad **`MODEL`**

Documentar una entidad `MODEL` implica crear y administrar instancias de `DocumentPart` para el modelo y para todas las `properties` del modelo. Por ejemplo, el modelo `Error` integrado con cada API de manera predeterminada tiene la siguiente definición de esquema

```
{
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "Error Schema",
  "type" : "object",
  "properties" : {
    "message" : { "type" : "string" }
  }
}
```

y requiere dos instancias de `DocumentationPart`, una para `Model` y otra para su propiedad `message`:

```
{
  "location": {
    "type": "MODEL",
    "name": "Error"
```

```
  },
  "properties": {
    "title": "Error Schema",
    "description": "A description of the Error model"
  }
}
```

Protección de los datos

```
{
  "location": {
    "type": "MODEL",
    "name": "Error.message"
  },
  "properties": {
    "description": "An error message."
  }
}
```

Cuando se exporta la API, las propiedades de `DocumentationPart` invalidarán los valores del esquema original.

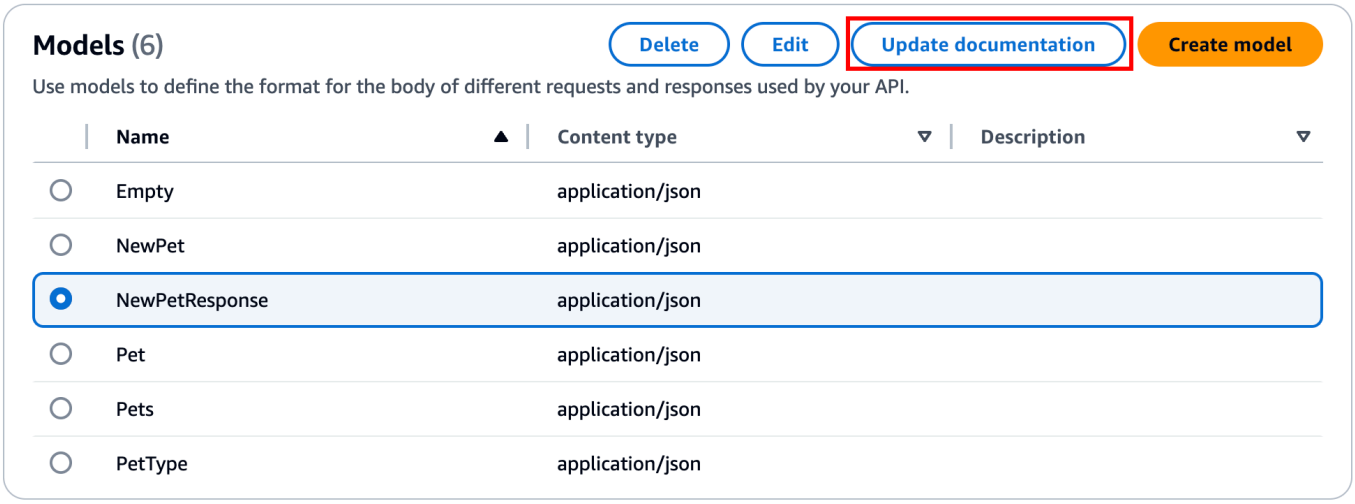
Para agregar una nueva pieza de documentación para una entidad `MODEL`, haga lo siguiente:

1. En el panel de navegación principal, elija Documentación y, a continuación, elija Crear pieza de documentación.
2. En Tipo de documentación, seleccione Modelo.
3. En Nombre, ingrese un nombre para el modelo.
4. Ingrese una descripción en el editor de texto.
5. Elija Crear pieza de documentación. Puede crear documentación para modelos que no figuren en la lista.
6. Si es necesario, repita estos pasos para añadir o editar una pieza de documentación para otros modelos.

Para agregar una nueva pieza de documentación para una entidad `MODEL` en el panel Modelos, haga lo siguiente:

1. En el panel de navegación principal, elija Modelos.

2. Elija el modelo y, a continuación, elija Actualizar documentación.



Para editar una pieza existente de la documentación, haga lo siguiente:

1. En el panel Documentación, elija la pestaña Modelos.
2. Utilice la barra de búsqueda o seleccione el modelo y, a continuación, elija Editar.

Documentar una entidad **`AUTHORIZER`**

Para agregar una nueva pieza de documentación para una entidad `AUTHORIZER`, haga lo siguiente:

1. En el panel de navegación principal, elija Documentación y, a continuación, elija Crear pieza de documentación.
2. En Tipo de documentación, seleccione Autorizador.
3. En Nombre, ingrese el nombre del autorizador.
4. Ingrese una descripción en el editor de texto. Especifique un valor para el campo válido de `location` para el autorizador.
5. Elija Crear pieza de documentación. Puede crear documentación para autorizadores que no figuren en la lista.
6. Si es necesario, repita estos pasos para añadir o editar una pieza de documentación para otros autorizadores.

Para editar una pieza existente de la documentación, haga lo siguiente:

1. En el panel Documentación, elija la pestaña Autorizadores.
2. Utilice la barra de búsqueda o seleccione el autorizador y, a continuación, elija Editar.

Publicación de documentación de la API mediante la consola de API Gateway

El procedimiento siguiente describe cómo publicar una versión de la documentación.

Para publicar una versión de la documentación mediante la consola de API Gateway

1. En el panel de navegación principal, elija Documentación.
2. Elija Publicar documentación.
3. Configure la publicación:
    a. En Etapa, seleccione una etapa.
    b. En Versión, ingrese un identificador de versión, por ejemplo, `1.0.0`.
    c. (Opcional) En Description (Descripción), introduzca una descripción.
4. Elija Publish.

Ahora puede empezar a descargar la documentación publicada exportando la documentación a un archivo de OpenAPI externo. Para obtener más información, consulte the section called “Exportación de una API de REST”.

Proceso de documentación de una API mediante la API de REST de API Gateway

En esta sección se describe cómo crear y mantener piezas de documentación de una API mediante la API de REST de API Gateway.

Antes de crear y editar la documentación de una API, primero debe crear la API. En esta sección, usaremos la API PetStore como ejemplo. Para crear una API mediante la consola de API Gateway, siga las instrucciones de Tutorial: Crear una API de REST importando un ejemplo.

Temas

- Documentar la entidad API
- Documentar una entidad RESOURCE
- Documentar una entidad METHOD
- Documentar una entidad QUERY_PARAMETER

- Documentar una entidad PATH_PARAMETER
- Documentar una entidad REQUEST_BODY
- Documentar una entidad REQUEST_HEADER
- Documentar una entidad RESPONSE
- Documentar una entidad RESPONSE_HEADER
- Documentar una entidad AUTHORIZER
- Documentar una entidad MODEL
- Actualización de las piezas de documentación
- Mostrar piezas de documentación

Documentar la entidad **API**

Si desea agregar documentación sobre una API, agregue un recurso DocumentationPart a la entidad de API:

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "API"
    },
    "properties": "{\n\t\"info\": {\n\t\t\"description\" : \"Your first API with Amazon
 API Gateway.\"\n\t}\n}"
}
```

Si es correcta, la operación devuelve una respuesta `201 Created` que contiene la instancia `DocumentationPart` recién creada en la carga. Por ejemplo:

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

Si ya se ha agregado la pieza de documentación, se devuelve una respuesta `409 Conflict`, que contiene el mensaje de error `Documentation part already exists for the specified location: type 'API'.`". En este caso, debe llamar a la operación documentationpart:update.

```
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}"
  } ]
}
```

La respuesta correcta devuelve un código de estado `200 OK` con la carga que contiene la instancia `DocumentationPart` actualizada en la carga.

Documentar una entidad **`RESOURCE`**

Para agregar documentación para el recurso raíz de una API, agregue el recurso DocumentationPart de destino para el recurso Resource correspondiente:

```
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}"
}
```

Si es correcta, la operación devuelve una respuesta `201 Created` que contiene la instancia `DocumentationPart` recién creada en la carga. Por ejemplo:

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

Cuando no se especifica la ruta del recurso, se considera que el recurso es el recurso raíz. Puede añadir `"path": "/"` a `properties` para que la especificación sea explícita.

Para crear documentación para un recurso secundario de una API, agregue el recurso DocumentationPart de destino para el recurso Resource correspondiente:

```
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}"
}
```

Si es correcta, la operación devuelve una respuesta `201 Created` que contiene la instancia `DocumentationPart` recién creada en la carga. Por ejemplo:

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

Para agregar documentación para un recurso secundario especificado por un parámetro de ruta, agregue un recurso DocumentationPart de destino para el recurso Resource correspondiente:

```
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}"
}
```

Si es correcta, la operación devuelve una respuesta `201 Created` que contiene la instancia `DocumentationPart` recién creada en la carga. Por ejemplo:

```
{
  "_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**
>
> La instancia de DocumentationPart de una entidad `RESOURCE` no la pueden heredar sus recursos secundarios.

Documentar una entidad **`METHOD`**

Para agregar documentación para un método de una API, agregue un recurso DocumentationPart de destino para el recurso Method correspondiente:

```
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}"
}
```

Si es correcta, la operación devuelve una respuesta `201 Created` que contiene la instancia `DocumentationPart` recién creada en la carga. Por ejemplo:

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

Si es correcta, la operación devuelve una respuesta `201 Created` que contiene la instancia `DocumentationPart` recién creada en la carga. Por ejemplo:

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

Si no se especifica el campo `location.method` en la solicitud anterior, se supone que se trata del método `ANY`, representado por un carácter comodín: `*`.

Para actualizar el contenido de documentación de una entidad `METHOD`, llame a la operación documentationpart:update, proporcionando un nuevo mapa `properties`:

```
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}"
  } ]
}
```

La respuesta correcta devuelve un código de estado `200 OK` con la carga que contiene la instancia `DocumentationPart` actualizada en la carga. Por ejemplo:

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

Documentar una entidad **QUERY_PARAMETER**

Para agregar documentación para un parámetro de consulta de solicitud, agregue un recurso DocumentationPart de destino para el tipo `QUERY_PARAMETER`, con los campos válidos `path` y `name`.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "location" : {
         "type" : "QUERY_PARAMETER",
         "path" : "/pets",
         "method" : "GET",
         "name" : "page"
    },
    "properties": "{\n\t\"description\" : \"Page number of results to return.\"\n}"
}
```

Si es correcta, la operación devuelve una respuesta `201 Created` que contiene la instancia `DocumentationPart` recién creada en la carga. Por ejemplo:

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

El mapa `properties` de la pieza de documentación de una entidad `QUERY_PARAMETER` lo pueden heredar alguna de sus entidades `QUERY_PARAMETER` secundarias. Por ejemplo, si añade un recurso `treats` detrás de `/pets/{petId}`, habilita el método `GET` en `/pets/{petId}/treats` y expone el parámetro de consulta `page`, este parámetro de consulta secundario hereda el mapa `DocumentationPart` de `properties` desde el parámetro de consulta de igual nombre del método `GET /pets`, a menos que añada explícitamente un recurso `DocumentationPart` al parámetro de consulta `page` del método `GET /pets/{petId}/treats`.

Documentar una entidad **`PATH_PARAMETER`**

Para agregar documentación para un parámetro de ruta, agregue un recurso DocumentationPart a la entidad `PATH_PARAMETER`.

```
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}"
```

```
}
```

Si es correcta, la operación devuelve una respuesta `201 Created` que contiene la instancia `DocumentationPart` recién creada en la carga. Por ejemplo:

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

Documentar una entidad **REQUEST_BODY**

Para agregar documentación para un cuerpo de solicitud, agregue un recurso DocumentationPart al cuerpo de solicitud.

```
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}"
}
```

Si es correcta, la operación devuelve una respuesta `201 Created` que contiene la instancia `DocumentationPart` recién creada en la carga. Por ejemplo:

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

Documentar una entidad **`REQUEST_HEADER`**

Para agregar documentación para un encabezado de solicitud, agregue un recurso DocumentationPart al encabezado de solicitud.

```
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}"
}
```

Si es correcta, la operación devuelve una respuesta `201 Created` que contiene la instancia `DocumentationPart` recién creada en la carga. Por ejemplo:

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

Documentar una entidad **`RESPONSE`**

Para agregar documentación para una respuesta de un código de estado, agregue un recurso DocumentationPart de destino al recurso MethodResponse correspondiente.

```
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}"
}
```

Si es correcta, la operación devuelve una respuesta `201 Created` que contiene la instancia `DocumentationPart` recién creada en la carga. Por ejemplo:

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

Documentar una entidad **RESPONSE_HEADER**

Para agregar documentación para un encabezado de respuesta, agregue un recurso DocumentationPart al encabezado de respuesta.

```
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}"
```

Si es correcta, la operación devuelve una respuesta `201 Created` que contiene la instancia `DocumentationPart` recién creada en la carga. Por ejemplo:

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

La documentación de este encabezado de respuesta `Content-Type` es la documentación predeterminada para los encabezados `Content-Type` de todas las respuestas de la API.

Documentar una entidad **`AUTHORIZER`**

Para agregar documentación para un autorizador de API, agregue un recurso DocumentationPart de destino al autorizador especificado.

```
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}"
}
```

Si es correcta, la operación devuelve una respuesta `201 Created` que contiene la instancia `DocumentationPart` recién creada en la carga. Por ejemplo:

```
{
  "_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**
>
> La instancia de DocumentationPart de una entidad `AUTHORIZER` no la pueden heredar sus recursos secundarios.

Documentar una entidad **`MODEL`**

Documentar una entidad `MODEL` implica crear y administrar instancias de `DocumentPart` para el modelo y para todas las `properties` del modelo. Por ejemplo, el modelo `Error` integrado con cada API de manera predeterminada tiene la siguiente definición de esquema

```
{
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "Error Schema",
  "type" : "object",
  "properties" : {
    "message" : { "type" : "string" }
  }
}
```

y requiere dos instancias de `DocumentationPart`, una para `Model` y otra para su propiedad `message`:

```
{
  "location": {
    "type": "MODEL",
    "name": "Error"
  },
  "properties": {
    "title": "Error Schema",
    "description": "A description of the Error model"
  }
}
```

Protección de los datos

```
{
  "location": {
    "type": "MODEL",
    "name": "Error.message"
  },
  "properties": {
    "description": "An error message."
  }
}
```

Cuando se exporta la API, las propiedades de `DocumentationPart` invalidarán los valores del esquema original.

Para agregar documentación para un modelo de API, agregue un recurso DocumentationPart de destino al modelo especificado.

```
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}"
}
```

Si es correcta, la operación devuelve una respuesta `201 Created` que contiene la instancia `DocumentationPart` recién creada en la carga. Por ejemplo:

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

Repita el mismo paso si desea crear una instancia de DocumentationPart para cualquiera de las propiedades del modelo.

> **Note**
>
> La instancia de DocumentationPart de una entidad `MODEL` no la pueden heredar sus recursos secundarios.

Actualización de las piezas de documentación

Para actualizar las piezas de documentación de cualquier tipo de entidad de API, envíe una solicitud PATCH en una instancia de DocumentationPart de un identificador de pieza especificado para sustituir el mapa `properties` existente por uno nuevo.

```
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"
  } ]
}
```

La respuesta correcta devuelve un código de estado `200 OK` con la carga que contiene la instancia `DocumentationPart` actualizada en la carga.

Puede actualizar varias piezas de documentación en una sola solicitud `PATCH`.

Mostrar piezas de documentación

Para mostrar las piezas de documentación de cualquier tipo de entidad de API, envíe una solicitud GET en una colección DocumentationParts.

```
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
```

La respuesta correcta devuelve un código de estado `200 OK` con la carga que contiene las instancias `DocumentationPart` disponibles en la carga.

Publicación de documentación de la API mediante la API de REST de API Gateway

Para publicar la documentación de una API, cree, actualice u obtenga una snapshot de documentación y después asóciela a una etapa de API. Cuando crea una snapshot de documentación, también puede asociarla a una etapa de API al mismo tiempo.

Temas

- Crear una snapshot de documentación y asociarla a una etapa de API
- Crear una snapshot de documentación
- Actualizar una snapshot de documentación
- Obtener una snapshot de documentación
- Asociar una snapshot de documentación con una etapa de API
- Descargar una snapshot de documentación asociada a una etapa

Crear una snapshot de documentación y asociarla a una etapa de API

Para crear una snapshot de piezas de documentación de una API y asociarla a una etapa de API al mismo tiempo, envíe la siguiente solicitud `POST`:

```
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"
}
```

Si es correcta, la operación devuelve una respuesta `200 OK` que contiene la instancia `DocumentationVersion` recién creada como la carga.

También puede crear una instantánea de documentación sin asociarla primero a una etapa de API y después llamar a restapi:update para asociar la instantánea a una etapa de API específica. También puede actualizar o consultar una snapshot de documentación existente y después actualizar su asociación de etapa. Le mostramos los pasos en las próximas cuatro secciones.

Crear una snapshot de documentación

Para crear una instantánea de piezas de documentación de una API, cree un nuevo recurso DocumentationVersion y agréguelo a la colección DocumentationVersions de la API:

```
POST /restapis/restapi_id/documentation/versions HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "documentationVersion" : "1.0.0",
    "description" : "My API Documentation v1.0.0"
}
```

Si es correcta, la operación devuelve una respuesta `200 OK` que contiene la instancia `DocumentationVersion` recién creada como la carga.

Actualizar una snapshot de documentación

Solo puede actualizar una instantánea de documentación modificando la propiedad `description` del recurso DocumentationVersion correspondiente. El siguiente ejemplo muestra cómo actualizar la descripción de la snapshot de documentación identificada por su identificador de versión, *`version`* (p. ej., `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."
    }]
}
```

Si es correcta, la operación devuelve una respuesta `200 OK` que contiene la instancia `DocumentationVersion` actualizada como la carga.

Obtener una snapshot de documentación

Para obtener una instantánea de documentación, envíe una solicitud GET al recurso DocumentationVersion especificado. El siguiente ejemplo muestra cómo obtener una snapshot de documentación de un identificador de versión determinado, 1.0.0.

```
GET /restapis/<restapi_id>/documentation/versions/1.0.0 HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret
```

Asociar una snapshot de documentación con una etapa de API

Para publicar la documentación de la API, asocie una snapshot de documentación a una etapa de API. Debe haber creado una etapa de API antes de asociar la versión de documentación a la etapa.

Para asociar una instantánea de documentación a una etapa de API, mediante la API de REST de API Gateway, llame a la operación stage:update para establecer la versión de documentación que desee en la propiedad `stage.documentationVersion`:

```
PATCH /restapis/RESTAPI_ID/stages/STAGE_NAME
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret

{
    "patchOperations": [{
        "op": "replace",
        "path": "/documentationVersion",
        "value": "VERSION_IDENTIFIER"
    }]
}
```

Descargar una snapshot de documentación asociada a una etapa

Una vez asociada una versión de las partes de la documentación a una etapa, puede exportar las partes de la documentación junto con las definiciones de la entidad de la API a un archivo externo mediante la consola de API Gateway, la API REST de API Gateway, uno de sus SDK o la AWS CLI para API Gateway. El proceso es el mismo que para exportar la API. El formato del archivo exportado puede ser JSON o YAML.

Mediante la API de REST de API Gateway, también puede establecer explícitamente el parámetro de consulta `extension=documentation,integrations,authorizers` para incluir las piezas de documentación de la API, las integraciones de la API y los autorizadores en una exportación de la API. De forma predeterminada, se incluyen las piezas de documentación, pero se excluyen las integraciones y los autorizadores cuando se exporta una API. La salida predeterminada de la exportación de API es adecuada para la distribución de la documentación.

Para exportar la documentación de la API en un archivo JSON de OpenAPI externo mediante la API de REST de API Gateway, envíe la siguiente solicitud `GET`:

```
GET /restapis/restapi_id/stages/stage_name/exports/swagger?extensions=documentation
 HTTP/1.1
Accept: application/json
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: 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
```

Aquí, el objeto `x-amazon-apigateway-documentation` contiene las piezas de documentación y las definiciones de la entidad de API contienen las propiedades de documentación admitidas por OpenAPI. La salida no incluye los detalles de la integración ni de los autorizadores de Lambda (que anteriormente se denominaban autorizadores personalizados). Para incluir ambos detalles, establezca `extensions=integrations,authorizers,documentation`. Para incluir los detalles de las integraciones pero no de los autorizadores, establezca `extensions=integrations,documentation`.

Debe definir el encabezado `Accept:application/json` en la solicitud para mostrar el resultado en un archivo JSON. Para producir la salida en formato YAML, cambie el encabezado de la solicitud a `Accept:application/yaml`.

A modo de ejemplo, vamos a examinar una API que expone un método `GET` sencillo en el recurso raíz (`/`). Esta API tiene cuatro entidades de API definidas en un archivo de definición de OpenAPI, una para cada uno de los tipos `API`, `MODEL`, `METHOD` y `RESPONSE`. Se ha añadido una pieza de documentación a cada una de las entidades `API`, `METHOD` y `RESPONSE`. Al llamar al comando de exportación de la documentación anterior, obtenemos el siguiente resultado, con las piezas de documentación mostradas dentro del objeto `x-amazon-apigateway-documentation` como una extensión a un archivo de OpenAPI estándar.

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"
}
```

Para un atributo compatible con OpenAPI definido en el mapa `properties` de una pieza de documentación, API Gateway inserta el atributo en la definición de la entidad de API asociada. Un atributo de `x-`*`something`* es una extensión de OpenAPI estándar. Esta extensión se propaga a la definición de la entidad de API. Véase, por ejemplo, el atributo `x-example` del método `GET`. Un atributo como `foo` no forma parte de la especificación de OpenAPI y no se incluye en sus definiciones de entidad de API asociadas.

Si una herramienta de representación de la documentación (por ejemplo, la interfaz de usuario de OpenAPI) analiza las definiciones de la entidad de API para extraer los atributos de documentación,

ninguno de los atributos `properties` que no sean compatibles con OpenAPI de una instancia de `DocumentationPart` estará disponible para la herramienta. Sin embargo, si una herramienta de representación de documentación analiza el objeto `x-amazon-apigateway-documentation` para obtener contenido, o si la herramienta llama a restapi:documentation-parts y documenationpart:by-id para recuperar piezas de documentación de API Gateway, la herramienta podrá mostrar todos los atributos de la documentación.

Para exportar la documentación con definiciones de entidad de API que contengan detalles de la integración en un archivo JSON de OpenAPI, envíe la siguiente solicitud `GET`:

```
GET /restapis/restapi_id/stages/stage_name/exports/swagger?
extensions=integrations,documentation HTTP/1.1
Accept: application/json
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret
```

Para exportar la documentación con definiciones de entidad de API que contengan detalles de las integraciones y autorizadores a un archivo YAML de OpenAPI, envíe la siguiente solicitud `GET`:

```
GET /restapis/restapi_id/stages/stage_name/exports/swagger?
extensions=integrations,authorizers,documentation HTTP/1.1
Accept: application/yaml
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
 Signature=sigv4_secret
```

Para utilizar la consola de API Gateway para exportar y descargar la documentación publicada de una API, siga las instrucciones de Exportar API REST con la consola de API Gateway.

Importar la documentación de API

De igual modo que para importar definiciones de entidad de API, puede importar piezas de documentación de un archivo de OpenAPI externo a una API mediante API Gateway. Las piezas de

documentación que se van a importar se especifican en la extensión Objeto x-amazon-apigateway-documentation en un archivo de definición de OpenAPI válido. La importación de la documentación no modifica las definiciones de entidad de API existentes.

Tiene la opción de combinar las piezas de documentación recién especificadas en piezas de documentación existentes en API Gateway o de sobrescribir las piezas de documentación existentes. En el modo `MERGE`, una nueva pieza de documentación definida en el archivo de OpenAPI se añade a la colección `DocumentationParts` de la API. Si ya existe un elemento `DocumentationPart` importado, el atributo importado reemplaza al existente si los dos son diferentes. Todos los demás atributos de documentación existentes permanecen tal como están. En el modo `OVERWRITE`, toda la colección `DocumentationParts` se reemplaza de acuerdo con el archivo de definición de OpenAPI importado.

Importar piezas de documentación mediante la API de REST de API Gateway

Para importar la documentación de API mediante la API de REST de API Gateway, llame a la operación documentationpart:import. En el siguiente ejemplo, se muestra cómo sobrescribir las piezas de documentación existentes de una API con un único método `GET /` , que devuelve una respuesta `200 OK` si se ejecuta correctamente.

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"
        }
      }
    ]
  }
}
```

Cuando se ejecuta correctamente, esta solicitud devuelve una respuesta 200 OK que contiene el elemento `DocumentationPartId` importado en la carga.

```
{
  "ids": [
    "kg3mth",
```

```
    "796rtf",
    "zhek4p",
    "5ukm9s"
  ]
}
```

Además, también puede llamar a restapi:import o restapi:put, proporcionando las piezas de documentación en el objeto `x-amazon-apigateway-documentation` como parte del archivo de OpenAPI de entrada de la definición de la API. Para excluir piezas de documentación de la importación de API, establezca `ignore=documentation` en los parámetros de consulta de la solicitud.

Importar piezas de documentación mediante la consola de API Gateway

Las siguientes instrucciones describen cómo importar piezas de documentación.

Para utilizar la consola para importar piezas de documentación de una API desde un archivo externo

1. En el panel de navegación principal, elija Documentación.
2. Seleccione Importar.
3. Si ya tiene documentación, seleccione Sobrescribir o Fusionar la nueva documentación.
4. Elija Seleccionar archivo para cargar un archivo de una unidad, o ingrese el contenido del archivo en la vista de archivos. Para ver un ejemplo, consulte la carga de la solicitud de ejemplo de Importar piezas de documentación mediante la API de REST de API Gateway.
5. Elija cómo gestionar las advertencias durante la importación. Seleccione Error en advertencias o Ignorar advertencias. Para obtener más información, consulte the section called “Errores y advertencias al importar la API a API Gateway”.
6. Seleccione Importar.

Controlar el acceso a la documentación de la API en API Gateway

Si tiene un equipo de documentación dedicado a escribir y editar la documentación de la API, puede configurar permisos de acceso distintos para los desarrolladores (para el desarrollo de la API) y para los redactores o editores (para el desarrollo de contenido). Esto resulta especialmente útil para un proveedor externo encargado de crear la documentación por usted.

Para otorgar al equipo de documentación acceso para crear, actualizar y publicar la documentación de la API, puede asignar un rol de IAM al equipo de documentación con la siguiente política de IAM, donde *account_id* es el ID de la cuenta de AWS del equipo de documentación.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [

    {
      "Sid": "StmtDocPartsAddEditViewDelete",
      "Effect": "Allow",
      "Action": [
        "apigateway:GET",
        "apigateway:PUT",
        "apigateway:POST",
        "apigateway:PATCH",
        "apigateway:DELETE"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1:111111111111:/restapis/*/documentation/*"
      ]
    }
  ]
}
```

Para obtener información sobre la configuración de permisos para obtener acceso a recursos de API Gateway, consulte the section called “Cómo funciona Amazon API Gateway con IAM”.

Generación de SDK para las API de REST en API Gateway

Para llamar a la API de REST de una forma específica para una plataforma o un lenguaje, debe generar el SDK específico de la plataforma o del lenguaje de la API. Se genera el SDK después de crear, probar e implementar la API en una etapa. Actualmente, API Gateway permite generar un SDK para una API en Java, JavaScript, Java para Android y Objective-C o Swift para iOS y Ruby.

En esta sección se explica cómo generar un SDK de una API de API Gateway. También muestra cómo usar el SDK generado en una aplicación Java, una aplicación Java para Android, Objective-C y Swift para iOS y una aplicación JavaScript.

Para facilitar la explicación, usaremos esta API de API Gateway, que expone esta función de Lambda de calculadora sencilla.

Antes de continuar, cree o importe la API e impleméntela al menos una vez en API Gateway. Para obtener instrucciones, consulte Implementación de las API de REST en API Gateway.

Temas

- Función de Lambda de calculadora sencilla
- API de calculadora sencilla en API Gateway
- Definición de OpenAPI de la API de calculadora sencilla
- Generación del SDK de Java de una API en API Gateway
- Para generar el SDK de Android de una API en API Gateway
- Generación del SDK de iOS de una API en API Gateway
- Generación del SDK de JavaScript de una API de REST en API Gateway
- Generación del SDK de Ruby de una API en API Gateway
- Generación de SDK para una API a través de los comandos de la AWS CLI en API Gateway

Función de Lambda de calculadora sencilla

A modo de ilustración, utilizaremos una función de Lambda de Node.js que realiza las operaciones binarias de suma, resta, multiplicación y división.

Temas

- Formato de entrada de la función de Lambda de calculadora sencilla
- Formato de salida de la función de Lambda de calculadora sencilla
- Implementación de la función de Lambda de calculadora sencilla

Formato de entrada de la función de Lambda de calculadora sencilla

Esta función toma una entrada con el formato siguiente:

```
{ "a": "Number", "b": "Number", "op": "string"}
```

donde op puede tener alguno de los valores siguientes: `(+, -, *, /, add, sub, mul, div)`.

Formato de salida de la función de Lambda de calculadora sencilla

Cuando una operación se realiza correctamente, devuelve el resultado en el siguiente formato:

```
{ "a": "Number", "b": "Number", "op": "string", "c": "Number"}
```

donde `c` contiene el resultado del cálculo.

Implementación de la función de Lambda de calculadora sencilla

La implementación de la función de Lambda es la siguiente:

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

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

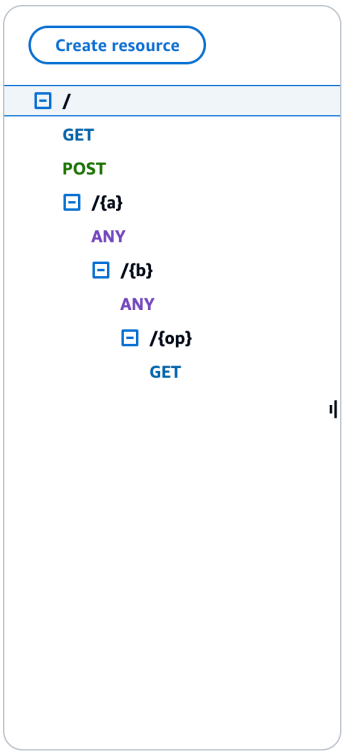
  const res = {};
  res.a = Number(event.a);
  res.b = Number(event.b);
  res.op = event.op;
  if (isNaN(event.a) || isNaN(event.b)) {
    return "400 Invalid Operand";
  }
  switch (event.op) {
    case "+":
    case "add":
      res.c = res.a + res.b;
      break;
    case "-":
    case "sub":
     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

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

  return res;
};
```

API de calculadora sencilla en API Gateway

Nuestra API de calculadora sencilla expone tres métodos (GET, POST, GET) para invocar the section called “Función de Lambda de calculadora sencilla”. A continuación, se muestra una representación gráfica de esta API:



Estos tres métodos muestran diferentes formas de proporcionar la entrada para la función de Lambda del backend con el objetivo de realizar la misma operación:

- El método `GET /?a=...&b=...&op=...` utiliza los parámetros de consulta para especificar la entrada.
- El método `POST /` utiliza una carga JSON `{"a":"Number", "b":"Number", "op":"string"}` para especificar la entrada.
- El método `GET /{a}/{b}/{op}` utiliza los parámetros de ruta para especificar la entrada.

Si no se define, API Gateway genera el nombre del método del SDK correspondiente combinando las partes de la ruta de acceso y el método HTTP. La parte de la ruta raíz (`/`) se denomina `Api Root`. Por ejemplo, el nombre predeterminado del SDK de Java para el método `GET /?a=...&b=...&op=...` de la API es `getABOp`, el del método `POST /` es `postApiRoot` y el del método `GET /{a}/{b}/{op}` es `getABOp`. Cada SDK puede personalizar la convención. Consulte en la documentación del código fuente del SDK generado los nombres de los métodos específicos del SDK.

Puede y debe anular los nombres de método del SDK predeterminados especificando la propiedad operationName en cada método de la API. Puede hacerlo al crear el método de la API o al actualizar el método de la API con la API de REST de API Gateway. En la definición de Swagger de la API, puede definir `operationId` para conseguir el mismo resultado.

Antes de mostrar cómo llamar a estos métodos con un SDK generado por API Gateway para esta API, vamos a recordar brevemente cómo se configuran. Para obtener instrucciones detalladas, consulte Desarrollo de las API de REST en API Gateway. Si es nuevo en API Gateway, consulte Elección de un tutorial de integración de AWS Lambda primero.

Crear modelos para la entrada y la salida

Para especificar una entrada con establecimiento inflexible de tipos en el SDK, creamos un modelo `Input` para la API. Para describir el tipo de datos del cuerpo de la respuesta, creamos un modelo `Output` y un modelo `Result`.

Para crear modelos para la entrada, la salida y el resultado

1. En el panel de navegación principal, elija Modelos.
2. Seleccione Crear modelo.
3. En Nombre, escriba **`input`**.
4. En Tipo de contenido, ingrese **`application/json`**.

Si no se encuentra ningún tipo de contenido coincidente, no se realiza la validación de la solicitud. Para utilizar el mismo modelo independientemente del tipo de contenido, introduzca **`$default`**.

5. En Esquema del modelo, escriba el siguiente modelo:

```
{
    "$schema" : "$schema": "http://json-schema.org/draft-04/schema#",
    "type":"object",
    "properties":{
        "a":{"type":"number"},
        "b":{"type":"number"},
        "op":{"type":"string"}
    },
    "title":"Input"
}
```

6. Seleccione Crear modelo.
7. Repita los pasos siguientes para crear un modelo `Output` y un modelo `Result`.

   Para el modelo `Output`, escriba lo siguiente en Esquema del modelo:

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type": "object",
    "properties": {
        "c": {"type":"number"}
    },
    "title": "Output"
}
```

   Para el modelo `Result`, escriba lo siguiente en Esquema del modelo. Sustituya el ID `abc123` de la API por su ID de API.

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type":"object",
    "properties":{
        "input":{
            "$ref":"https://apigateway.amazonaws.com/restapis/abc123/models/Input"
        },
```

```
        "output":{
            "$ref":"https://apigateway.amazonaws.com/restapis/abc123/models/Output"
        }
    },
    "title":"Result"
}
```

Configurar parámetros de consulta del método GET /

Para el método `GET /?a=..&b=..&op=..`, los parámetros de consulta se declaran en Method Request (Solicitud de método):

Para configurar los parámetros de la cadena de consulta GET/URL

1. En la sección Solicitud de método del método `GET` del recurso raíz (`/`), seleccione Editar.
2. Elija Parámetros de cadenas de consulta de URL y haga lo siguiente:

   a. Elija Añadir cadena de consulta.

   b. En Nombre, escriba **a**.

   c. Mantenga desactivados Obligatorio y Almacenamiento en caché.

   d. Mantenga Almacenamiento en caché desactivado.

   Repita los mismos pasos y cree una cadena de consulta llamada **b** y una cadena de consulta llamada **op**.
3. Seleccione Save.

Configuración del modelo de datos de la carga como entrada del backend

Para el método `POST /`, creamos el modelo `Input` y lo añadimos a la solicitud de método para definir la forma de los datos de entrada.

Para configurar el modelo de datos de la carga como entrada del backend

1. En la sección Solicitud de método del método `POST` del recurso raíz (`/`), seleccione Editar.
2. Elija Cuerpo de la solicitud.
3. Elija Add model (Añadir modelo).
4. En Tipo de contenido, ingrese **`application/json`**.

5. En Modelo, seleccione Entrada.
6. Seleccione Save.

Con este modelo, los clientes de la API pueden llamar al SDK para especificar la entrada creando una instancia del objeto `Input`. Sin este modelo, los clientes tendrán que crear un objeto de diccionario que represente la entrada JSON de la función de Lambda.

Configuración del modelo de datos de la salida del resultado desde el backend

Para los tres métodos, creamos el modelo `Result` y lo agregamos al elemento `Method Response` del método para definir la forma de la salida devuelta por la función de Lambda.

Para configurar el modelo de datos de la salida del resultado desde el backend

1. Seleccione el recurso /{a}/{b}/{op} y, a continuación, elija el método GET.
2. En la pestaña Respuesta del método, en Respuesta 200, elija Editar.
3. En Cuerpo de la respuesta, seleccione Agregar modelo.
4. En Tipo de contenido, ingrese **`application/json`**.
5. En Modelo, seleccione Resultado.
6. Seleccione Save.

Con este modelo, los clientes de la API pueden analizar la salida leyendo las propiedades de un objeto `Result`. Sin este modelo, los clientes tendrían que crear un objeto de diccionario que representara la salida JSON.

Definición de OpenAPI de la API de calculadora sencilla

A continuación, se muestra la definición de OpenAPI de la API de calculadora sencilla. Puede importarla en su cuenta. Sin embargo, debe restablecer los permisos basados en los recursos en la función de Lambda después de la importación. Para ello, vuelva a seleccionar la función de Lambda que ha creado en su cuenta desde Integration Request (Solicitud de integración) en la consola de API Gateway. Esto hará que la consola de API Gateway restablezca los permisos necesarios. También puede utilizar AWS Command Line Interface para el comando add-permission de Lambda.

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"
          }
        }
      }
    }
  }
}
```

Generación del SDK de Java de una API en API Gateway

En el siguiente procedimiento se muestra cómo generar el SDK de Java de una API en API Gateway.

Para generar el SDK de Java de una API en API Gateway

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST.
3. Elija Stages (Etapas).
4. En el panel Etapas, seleccione el nombre de la etapa.
5. Abra el menú Acciones de etapa y, a continuación, elija Generar SDK.
6. En Plataforma, elija la plataforma Java y haga lo siguiente:

    a. En Service Name (Nombre del servicio), especifique el nombre de su SDK. Por ejemplo, **`SimpleCalcSdk`**. Se convertirá en el nombre de la clase del cliente de SDK. El nombre se corresponde con la etiqueta `<name>` en `<project>` en el archivo pom.xml, que se encuentra en la carpeta del proyecto del SDK. No incluya guiones.

    b. En Java Package Name (Nombre del paquete Java), especifique un nombre de paquete para el SDK. Por ejemplo, **`examples.aws.apig.simpleCalc.sdk`**. Este nombre

de paquete se utiliza como el espacio de nombres de su biblioteca de SDK. No incluya guiones.

c. En Java Build System (Sistema de compilación de Java), escriba **`maven`** o **`gradle`** para especificar el sistema de compilación.

d. En Java Group Id (ID de grupo de Java), escriba un identificador de grupo para el proyecto del SDK. Por ejemplo, escriba **`my-apig-api-examples`**. Este identificador se corresponde con la etiqueta `<groupId>` en `<project>` en el archivo `pom.xml`, que está en la carpeta del proyecto del SDK.

e. En Java Artifact Id (ID de artefacto de Java), escriba un identificador de artefacto para el proyecto del SDK. Por ejemplo, escriba **`simple-calc-sdk`**. Este identificador se corresponde con la etiqueta `<artifactId>` en `<project>` en el archivo `pom.xml`, que está en la carpeta del proyecto del SDK.

f. En Java Artifact Version (Versión del artefacto de Java), escriba una cadena de identificador de la versión. Por ejemplo, **`1.0.0`**. Este identificador de versión se corresponde con la etiqueta `<version>` en `<project>` en el archivo `pom.xml`, que está en la carpeta del proyecto del SDK.

g. En Source Code License Text (Texto de licencia de código fuente), escriba el texto de la licencia del código fuente, si procede.

7. Elija Generate SDK (Generar SDK) y, a continuación, siga las instrucciones que aparecen en pantalla para descargar el SDK generado por API Gateway.

Siga las instrucciones de Uso de un SDK de Java generado por API Gateway para una API REST para utilizar el SDK generado.

Cada vez que actualiza una API, tiene que volver a implementarla y generar de nuevo el SDK para que las actualizaciones estén incluidas.

Para generar el SDK de Android de una API en API Gateway

En el siguiente procedimiento se muestra cómo generar el SDK de Android de una API en API Gateway.

Para generar el SDK de Android de una API en API Gateway

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST.

3. Elija Stages (Etapas).
4. En el panel Etapas, seleccione el nombre de la etapa.
5. Abra el menú Acciones de etapa y, a continuación, elija Generar SDK.
6. En Plataforma, elija la plataforma Android y haga lo siguiente:

    a. En Group ID (ID de grupo), escriba el identificador único para el proyecto correspondiente. Este identificador se utiliza en el archivo `pom.xml` (por ejemplo, **`com.mycompany`**).

    b. En Invoker package (Paquete de invocador), especifique el espacio de nombres para las clases del cliente generadas (por ejemplo, **`com.mycompany.clientsdk`**).

    c. En Artifact ID (ID de artefacto), especifique el nombre del archivo .jar compilado sin la versión. Este identificador se utiliza en el archivo `pom.xml` (por ejemplo, **`aws-apigateway-api-sdk`**).

    d. En Artifact version (Versión de artefacto), escriba el número de versión del artefacto para el cliente generado. Este nombre se utiliza en el archivo `pom.xml` y debe seguir el patrón *`versión principal`*.*`versión secundaria`*.*`parche`* (por ejemplo, **`1.0.0`**).
7. Elija Generate SDK (Generar SDK) y, a continuación, siga las instrucciones que aparecen en pantalla para descargar el SDK generado por API Gateway.

Siga las instrucciones de Uso de un SDK de Android generado por API Gateway para una API REST para utilizar el SDK generado.

Cada vez que actualiza una API, tiene que volver a implementarla y generar de nuevo el SDK para que las actualizaciones estén incluidas.

Generación del SDK de iOS de una API en API Gateway

En el siguiente procedimiento se muestra cómo generar el SDK de iOS de una API en API Gateway.

Para generar el SDK de iOS de una API en API Gateway

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST.
3. Elija Stages (Etapas).
4. En el panel Etapas, seleccione el nombre de la etapa.
5. Abra el menú Acciones de etapa y, a continuación, elija Generar SDK.
6. En Plataforma, elija la plataforma iOS (Objective-C) o iOS (Swift) y haga lo siguiente:

- Escriba un prefijo único en el cuadro Prefix (Prefijo).

  El efecto del prefijo es el siguiente: si asigna, por ejemplo, **`SIMPLE_CALC`** como el prefijo para el SDK de la API SimpleCalc con los modelos `input`, `output` y `result`, el SDK generado contendrá la clase `SIMPLE_CALCSimpleCalcClient` que encapsula la API, incluidas las solicitudes y las respuestas del método. Además, el SDK generado contendrá las clases `SIMPLE_CALCinput`, `SIMPLE_CALCoutput` y `SIMPLE_CALCresult` para representar la entrada, la salida y los resultados, respectivamente, para representar la entrada y la salida de la solicitud. Para obtener más información, consulte Uso de un SDK de iOS generado por API Gateway para una API REST en Objective-C o Swift.

7. Elija Generate SDK (Generar SDK) y, a continuación, siga las instrucciones que aparecen en pantalla para descargar el SDK generado por API Gateway.

Siga las instrucciones de Uso de un SDK de iOS generado por API Gateway para una API REST en Objective-C o Swift para utilizar el SDK generado.

Cada vez que actualiza una API, tiene que volver a implementarla y generar de nuevo el SDK para que las actualizaciones estén incluidas.

Generación del SDK de JavaScript de una API de REST en API Gateway

En el siguiente procedimiento se muestra cómo generar el SDK de JavaScript de una API en API Gateway.

Para generar el SDK de JavaScript de una API en API Gateway

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST.
3. Elija Stages (Etapas).
4. En el panel Etapas, seleccione el nombre de la etapa.
5. Abra el menú Acciones de etapa y, a continuación, elija Generar SDK.
6. En Plataforma, elija la plataforma JavaScript.
7. Elija Generate SDK (Generar SDK) y, a continuación, siga las instrucciones que aparecen en pantalla para descargar el SDK generado por API Gateway.

Siga las instrucciones de Uso de un SDK de JavaScript generado por API Gateway para una API REST para utilizar el SDK generado.

Cada vez que actualiza una API, tiene que volver a implementarla y generar de nuevo el SDK para que las actualizaciones estén incluidas.

Generación del SDK de Ruby de una API en API Gateway

En el siguiente procedimiento se muestra cómo generar el SDK de Ruby de una API en API Gateway.

Para generar el SDK de Ruby de una API en API Gateway

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST.
3. Elija Stages (Etapas).
4. En el panel Etapas, seleccione el nombre de la etapa.
5. Abra el menú Acciones de etapa y, a continuación, elija Generar SDK.
6. En Plataforma, elija la plataforma Ruby y haga lo siguiente:
    a. En Service Name (Nombre del servicio), especifique el nombre de su SDK. Por ejemplo, **`SimpleCalc`**. Esto se utiliza para generar el espacio de nombres de la gema de Ruby de la API. El nombre debe contener únicamente letras, (`a-zA-Z`); no debe incluir números ni caracteres especiales.
    b. En Ruby Gem Name (Nombre de gema de Ruby), especifique el nombre de la gema de Ruby que contiene el código fuente del SDK generado para la API. De forma predeterminada, el valor especificado será el nombre de servicio en minúsculas con el sufijo `-sdk`; por ejemplo, **`simplecalc-sdk`**.
    c. En Ruby Gem Version (Versión de gema de Ruby), especifique el número de versión de la gema de Ruby generada. De forma predeterminada, está establecido en `1.0.0`.
7. Elija Generate SDK (Generar SDK) y, a continuación, siga las instrucciones que aparecen en pantalla para descargar el SDK generado por API Gateway.

Siga las instrucciones de Uso de un SDK de Ruby generado por API Gateway para una API REST para utilizar el SDK generado.

Cada vez que actualiza una API, tiene que volver a implementarla y generar de nuevo el SDK para que las actualizaciones estén incluidas.

Generación de SDK para una API a través de los comandos de la AWS CLI en API Gateway

Si desea utilizar la AWS CLI para generar y descargar un SDK de una API para una plataforma compatible, llame al comando get-sdk. A continuación, veremos este comando en algunas de las plataformas compatibles.

Temas

- Creación y descarga del SDK de Java para Android a través de la AWS CLI
- Creación y descarga del SDK de JavaScript a través de la AWS CLI
- Creación y descarga del SDK de Ruby a través de la AWS CLI

Creación y descarga del SDK de Java para Android a través de la AWS CLI

Para generar y descargar un SDK de Java para Android creado por API Gateway a partir de una API (`udpuvvzbkc`) en una etapa determinada (`test`), llame al comando tal y como se indica a continuación:

```
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
```

La última entrada de `~/apps/myApi/myApi-android-sdk.zip` es la ruta de acceso al archivo del SDK descargado, `myApi-android-sdk.zip`.

Creación y descarga del SDK de JavaScript a través de la AWS CLI

Para generar y descargar un SDK de JavaScript creado por API Gateway a partir de una API (`udpuvvzbkc`) en una etapa determinada (`test`), llame al comando tal y como se indica a continuación:

```
aws apigateway get-sdk \
            --rest-api-id udpuvvzbkc \
            --stage-name test \
            --sdk-type javascript \
            ~/apps/myApi/myApi-js-sdk.zip
```

La última entrada de `~/apps/myApi/myApi-js-sdk.zip` es la ruta de acceso al archivo del SDK descargado, `myApi-js-sdk.zip`.

Creación y descarga del SDK de Ruby a través de la AWS CLI

Si desea generar y descargar un SDK de Ruby de una API (`udpuvvzbkc`) en una etapa determinada (`test`), llame al comando tal y como se indica a continuación:

```
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
```

La última entrada de `~/apps/myApi/myApi-ruby-sdk.zip` es la ruta de acceso al archivo del SDK descargado, `myApi-ruby-sdk.zip`.

A continuación, le mostramos cómo utilizar el SDK generado para llamar a la API subyacente. Para obtener más información, consulte Invocación de las API de REST en API Gateway.

Venda sus API de API Gateway a través de AWS Marketplace

Después de crear, probar e implementar las API, puede empaquetarlas en un plan de uso de API Gateway y venderlo como producto de software como servicio (SaaS) a través de AWS Marketplace. AWS Marketplace factura a los compradores de la API que se han suscrito a su oferta de producto en función del número de solicitudes realizadas en el plan de uso.

Para vender sus API en AWS Marketplace, debe configurar el canal de ventas para integrar AWS Marketplace en API Gateway. En términos generales, esto implica publicar el producto en AWS Marketplace, configurar un rol de IAM con las políticas adecuadas para permitir a API Gateway enviar métricas de uso a AWS Marketplace, asociar un producto de AWS Marketplace con un plan de uso de API Gateway y asociar un comprador de AWS Marketplace con una clave de API de API Gateway. En las próximas secciones se ofrece información adicional.

Para obtener más información sobre la venta de la API como un producto SaaS en AWS Marketplace, consulte la Guía del usuario de AWS Marketplace.

Temas

- Inicializar la integración de AWS Marketplace con API Gateway
- Administrar la suscripción de clientes a planes de uso

Inicializar la integración de AWS Marketplace con API Gateway

Las siguientes tareas son para inicializar por única vez la integración de AWS Marketplace con API Gateway, lo que le permite vender sus API como un producto de SaaS.

Publicar un producto en AWS Marketplace

Para publicar su plan de uso como un producto SaaS, envíe un formulario de carga de producto través de AWS Marketplace. El producto debe contener una dimensión con el nombre `apigateway` del tipo `requests`. Esta dimensión define el precio por solicitud y API Gateway la utiliza para medir las solicitudes a sus API.

Crear el rol de medición

Cree un rol de IAM denominado `ApiGatewayMarketplaceMeteringRole` con las siguientes políticas de ejecución y confianza. Este rol permite a API Gateway enviar métricas de uso a AWS Marketplace en su nombre.

Política de ejecución del rol de medición

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "aws-marketplace:BatchMeterUsage",
        "aws-marketplace:ResolveCustomer"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
```

```
  ]
}
```

Política de relación de confianza del rol de medición

JSON

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

Asociar el plan de uso con el producto de AWS Marketplace

Cuando publica un producto en AWS Marketplace, recibe un código de producto de AWS Marketplace. Para integrar API Gateway en AWS Marketplace, asocie su plan de uso con el código de producto de AWS Marketplace. Para habilitar la asociación mediante el establecimiento del campo `productCode` del `UsagePlan` de API Gateway en el código de producto de AWS Marketplace, puede utilizar la consola de API Gateway, la API de REST de API Gateway, la AWS CLI para API Gateway o un AWS SDK para API Gateway. En el ejemplo de código siguiente se utiliza la API de REST de API Gateway:

```
PATCH /usageplans/USAGE_PLAN_ID
Host: apigateway.region.amazonaws.com
Authorization: ...

{
    "patchOperations" : [{
        "path" : "/productCode",
        "value" : "MARKETPLACE_PRODUCT_CODE",
        "op" : "replace"
```

```
    }]
}
```

Administrar la suscripción de clientes a planes de uso

La aplicación del portal para desarrolladores se encarga de las siguientes tareas.

Cuando un cliente se suscribe a su producto mediante AWS Marketplace, AWS Marketplace reenvía una solicitud `POST` a la URL de suscripciones SaaS que registró cuando publicó su producto en AWS Marketplace. La solicitud `POST` incluye un parámetro `x-amzn-marketplace-token` que contiene la información del comprador. Siga las instrucciones de Incorporación de clientes de SaaS para gestionar esta redirección en su aplicación del portal de desarrolladores.

En respuesta a la solicitud de suscripción de un cliente, AWS Marketplace envía una notificación `subscribe-success` a un tema de Amazon SNS al que usted puede suscribirse. (Consulte Incorporación de clientes de SaaS). Para aceptar la solicitud de suscripción del cliente, administre la notificación `subscribe-success` mediante la creación o la recuperación de una clave de la API de API Gateway para el cliente, asociando el `customerId` aprovisionado por AWS Marketplace del cliente a las claves de la API y agregando después la clave de la API a su plan de uso.

Cuando se completa la solicitud de suscripción del cliente, la aplicación del portal para desarrolladores debe presentar al cliente la clave de API asociada e informarle de que debe incluirse en el encabezado `x-api-key` de las solicitudes a las API.

Cuando un cliente cancela una suscripción a un plan de uso, AWS Marketplace envía una notificación `unsubscribe-success` al tema de SNS. Para completar el proceso de anular la suscripción del cliente, administra la notificación `unsubscribe-success` eliminando las claves de API del cliente del plan de uso.

Autorizar a un cliente para obtener acceso a un plan de uso

Para autorizar el acceso a su plan de uso para un cliente determinado, utilice la API de API Gateway para recuperar o crear una clave de API para el cliente y agréguela al plan de uso.

En el siguiente ejemplo, se muestra cómo llamar a la API REST de API Gateway para crear una nueva clave de API con un valor de AWS Marketplace específico, `customerId` (*`MARKETPLACE_CUSTOMER_ID`*).

```
POST apikeys HTTP/1.1
Host: apigateway.region.amazonaws.com
Authorization: ...
```

```
{
  "name" : "my_api_key",
  "description" : "My API key",
  "enabled" : "false",
  "stageKeys" : [ {
    "restApiId" : "uycll6xg9a",
    "stageName" : "prod"
  } ],
  "customerId" : "MARKETPLACE_CUSTOMER_ID"
}
```

El siguiente ejemplo muestra cómo obtener una clave de API con un valor AWS Marketplace de `customerId` específico (*MARKETPLACE_CUSTOMER_ID*).

```
GET apikeys?customerId=MARKETPLACE_CUSTOMER_ID HTTP/1.1
Host: apigateway.region.amazonaws.com
Authorization: ...
```

Para agregar una clave de API a un plan de uso, cree un recurso `UsagePlanKey` con la clave de API para el plan de uso pertinente. El siguiente ejemplo muestra cómo realizar esto usando la API de REST de API Gateway, donde `n371pt` es el ID del plan de uso y `q5ugs7qjjh` es un `keyId` de API de ejemplo devuelto en los ejemplos anteriores.

```
POST /usageplans/n371pt/keys HTTP/1.1
Host: apigateway.region.amazonaws.com
Authorization: ...

{
    "keyId": "q5ugs7qjjh",
    "keyType": "API_KEY"
}
```

Asociar un cliente a una clave de API

Debe actualizar el campo de `ApiKey` `customerId` con el ID del cliente de AWS Marketplace. De esta forma, se asociará la clave de API con el cliente de AWS Marketplace, con lo que podrá medir y facturar las solicitudes del comprador. El siguiente ejemplo de código llama a la API de REST de API Gateway para hacerlo.

```
PATCH /apikeys/q5ugs7qjjh
```

```
Host: apigateway.region.amazonaws.com
Authorization: ...

{
    "patchOperations" : [{
        "path" : "/customerId",
        "value" : "MARKETPLACE_CUSTOMER_ID",
        "op" : "replace"
    }]
}
```

Protección de sus API de REST en API Gateway

API Gateway proporciona una serie de formas de proteger su API de ciertas amenazas, como usuarios malintencionados o picos de tráfico. Puede proteger su API usando estrategias como generar certificados SSL, configurar un firewall de aplicaciones web, establecer objetivos de limitación y permitir el acceso a su API desde una Virtual Private Cloud (VPC). En esta sección puede aprender cómo habilitar estas capacidades mediante API Gateway.

Temas

- Solicitud de certificados de cliente para la API con autenticación de TLS mutua en API Gateway
- Presentación de certificados de cliente a los servicios de backend con TLS mutuo en API Gateway
- Uso de AWS WAF para proteger sus API de REST en API Gateway
- Limitación de las solicitudes a sus API de REST para mejorar el rendimiento en API Gateway
- API de REST privadas en API Gateway

Solicitud de certificados de cliente para la API con autenticación de TLS mutua en API Gateway

La autenticación TLS mutua requiere la autenticación bidireccional entre el cliente y el servidor. Con la TLS mutua, los clientes deben presentar certificados X.509 para verificar su identidad y acceder a su API. La TLS mutua es un requisito frecuente para el Internet de las cosas (IoT) y las aplicaciones entre empresas.

Puede utilizar TLS mutua junto con otras operaciones de autorización y autenticación compatibles con API Gateway. API Gateway reenvía los certificados que los clientes proporcionan a los autorizadores de Lambda y a las integraciones del backend.

> **Important**
>
> De forma predeterminada, los clientes pueden invocar su API mediante el punto de conexión `execute-api` que API Gateway genera para su API. Para asegurarse de que los clientes solo puedan acceder a su API mediante un nombre de dominio personalizado con la TLS mutua, deshabilite el punto de conexión `execute-api` predeterminado. Para obtener más información, consulte the section called “Deshabilitación del punto de conexión predeterminado”.

Temas

- Requisitos previos de la TLS mutua
- Configuración de la TLS mutua para un nombre de dominio personalizado
- Invocar una API mediante un nombre de dominio personalizado que requiere la TLS mutua
- Actualización de su almacén de confianza
- Deshabilitar la TLS mutua
- Solución de problemas de TLS mutuo para su API de REST

Requisitos previos de la TLS mutua

Para configurar la TLS mutua necesita lo siguiente:

- Un nombre de dominio personalizado regional
- Al menos un certificado configurado en AWS Gestor de certificados para el nombre de dominio personalizado
- Un almacén de confianza configurado y cargado en Amazon S3

Nombres de dominio personalizados

Para habilitar la TLS mutua para una API REST, debe configurar un nombre de dominio personalizado para su API. Puede habilitar la TLS mutua para un nombre de dominio personalizado y, a continuación, proporcionar el nombre de dominio personalizado a los clientes. Para acceder a una API mediante un nombre de dominio personalizado que tenga habilitada la TLS mutua, los clientes deben presentar certificados en los que confíe en las solicitudes de API. Dispone de más información en the section called “Nombres de dominio personalizados”.

Uso de certificados emitidos por AWS Gestor de certificados

Puede solicitar un certificado de confianza pública directamente desde ACM o importar certificados públicos o autofirmados. Para configurar un certificado en ACM, vaya a ACM. Si desea importar un certificado, siga leyendo en la siguiente sección.

Uso de un certificado importado o de AWS Autoridad de certificación privada

Para utilizar un certificado importado en ACM o un certificado de AWS Autoridad de certificación privada con TLS mutua, API Gateway necesita un `ownershipVerificationCertificate` emitido por ACM. Este certificado de propiedad solo se utiliza para comprobar que dispone de permisos para utilizar el nombre de dominio. No se utiliza en el protocolo de enlace TLS. Si todavía no tiene un `ownershipVerificationCertificate`, vaya a https://console.aws.amazon.com/acm/ para configurarlo.

Este certificado deberá mantener su validez durante toda la vida útil del nombre de dominio. Si un certificado caduca y falla la renovación automática, se bloquearán todas las actualizaciones del nombre de dominio. Tendrá que actualizar el `ownershipVerificationCertificateArn` con un `ownershipVerificationCertificate` válido para poder realizar otros cambios. El `ownershipVerificationCertificate` no se puede utilizar como certificado de servidor para otro dominio de TLS mutua en API Gateway. Si un certificado se vuelve a importar directamente en ACM, el emisor debe ser el mismo.

Configuración del almacén de confianza

Los almacenes de confianza son archivos de texto cuya extensión es `.pem`. Son una lista de certificados de confianza procedentes de entidades de certificación. Para utilizar la TLS mutua, cree un almacén de confianza de certificados X.509 en los que confíe para acceder a su API.

En el almacén de confianza debe incluir la cadena de confianza completa, desde el certificado de entidad de certificación emisora hasta el certificado de entidad de certificación raíz. API Gateway acepta certificados de cliente emitidos por cualquier entidad de certificación presente en la cadena de confianza. Los certificados pueden ser de autoridades de certificación públicas o privadas. Los certificados pueden tener una longitud máxima de cadena de cuatro. También puede proporcionar certificados autofirmados. Se admiten los siguientes algoritmos en el almacén de confianza:

- SHA-256 o más seguro
- RSA-2048 o más seguro
- ECDSA-256 o ECDSA-384

API Gateway valida una serie de propiedades de certificado. Puede utilizar autorizadores de Lambda para realizar comprobaciones adicionales cuando un cliente invoque una API, tales como verificar si se ha revocado un certificado. API Gateway valida las siguientes propiedades:

Validación	Descripción
Sintaxis X.509	El certificado debe cumplir los requisitos de sintaxis X.509.
Integridad	El contenido del certificado no debe haberse alterado con respecto al firmado por la entidad de certificación del almacén de confianza.
Validez	El período de validez del certificado debe ser actual.
Encadenamiento de nombres/encadenamiento de claves	Los nombres y asuntos de los certificados deben formar una cadena ininterrumpida. Los certificados pueden tener una longitud máxima de cadena de cuatro.

Carga del almacén de confianza a un bucket de Amazon S3 en un solo archivo

El siguiente es un ejemplo del aspecto que puede tener un archivo .pem.

Example certificates.pem

```
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
...
```

El siguiente comando cp de la AWS CLI permite cargar `certificates.pem` en el bucket de Amazon S3.

```
aws s3 cp certificates.pem s3://bucket-name
```

Configuración de la TLS mutua para un nombre de dominio personalizado

Para configurar la TLS mutua para una API de REST, debe usar un nombre de dominio personalizado regional para la API, con una política de seguridad de TLS_1_2. Para obtener más información sobre cómo se elige una política de seguridad, consulte the section called “Elección de una política de seguridad”.

> **Note**
>
> La TLS mutua no es compatible con las API privadas.

Después de cargar su almacén de confianza en Amazon S3, puede configurar su nombre de dominio personalizado para que utilice TLS mutua. El siguiente comando create-domain-name permite crear un nombre de dominio personalizado con TLS mutua:

```
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
```

Después de crear el nombre de dominio, debe configurar los registros DNS y los mapeos de ruta base para las operaciones de la API. Para obtener más información, consulte Configuración de un nombre de dominio personalizado regional en API Gateway.

Invocar una API mediante un nombre de dominio personalizado que requiere la TLS mutua

Para invocar una API con la TLS mutua habilitada, los clientes deben presentar un certificado de confianza en la solicitud de la API. Cuando un cliente intenta invocar la API, API Gateway busca el emisor del certificado del cliente en el almacén de confianza. Para que API Gateway atienda la solicitud, en el almacén de confianza deben constar el emisor del certificado y la cadena de confianza completa hasta el certificado de entidad de certificación raíz.

El siguiente ejemplo de comando `curl` envía una solicitud a `api.example.com`, que incluye `my-cert.pem` en la solicitud. `my-key.key` es la clave privada del certificado.

```
curl -v --key ./my-key.key --cert ./my-cert.pem api.example.com
```

Su API solo se invoca si su almacén de confianza confía en el certificado. Las siguientes condiciones provocarán que API Gateway no pueda completar el protocolo de enlace TLS y deniegue la solicitud con un código de estado `403`. Si el certificado:

- no es de confianza
- ha caducado
- no utiliza un algoritmo compatible

> **Note**
>
> API Gateway no comprueba si se ha revocado un certificado.

Actualización de su almacén de confianza

Para actualizar los certificados del almacén de confianza, cargue un nuevo paquete de certificados en Amazon S3. Después, puede actualizar el nombre de dominio personalizado para que utilice el certificado actualizado.

Utilice el control de versiones de Amazon S3 para mantener varias versiones de su almacén de confianza. Cuando actualiza el nombre de dominio personalizado para utilizar una nueva versión del almacén de confianza, API Gateway devuelve advertencias si los certificados no son válidos.

API Gateway produce advertencias de certificado solo cuando actualiza el nombre de dominio. API Gateway no le notifica si caduca un certificado cargado anteriormente.

El siguiente comando update-domain-name permite actualizar un nombre de dominio personalizado para utilizar una nueva versión del almacén de confianza.

```
aws apigateway update-domain-name \
    --domain-name api.example.com \
    --patch-operations op='replace',path='/mutualTlsAuthentication/
truststoreVersion',value='abcdef123'
```

Deshabilitar la TLS mutua

El siguiente comando update-domain-name permite deshabilitar la TLS mutua:

```
aws apigateway update-domain-name \
    --domain-name api.example.com \
    --patch-operations op='replace',path='/mutualTlsAuthentication/
truststoreUri',value=''
```

Solución de problemas de TLS mutuo para su API de REST

A continuación se proporcionan consejos para solucionar errores y problemas que puedan surgir al activar TLS mutuo.

Solución de problemas de advertencias de certificados

Cuando se crea un nombre de dominio personalizado con TLS mutua, API Gateway devuelve advertencias si los certificados del almacén de confianza no son válidos. Esto también puede ocurrir cuando se actualiza un nombre de dominio personalizado para que utilice un nuevo almacén de confianza. Las advertencias indican el problema con el certificado y el asunto del certificado que produjo la advertencia. La TLS mutua aún está habilitada en la API, pero es posible que algunos clientes no puedan acceder a su API.

Debe descodificar los certificados del almacén de confianza para identificar cuál de ellos ha producido la advertencia. Puede utilizar herramientas como `openssl` para descodificar los certificados e identificar sus asuntos.

El siguiente comando muestra el contenido de un certificado, incluido su asunto:

```
openssl x509 -in certificate.crt -text -noout
```

Actualice o elimine los certificados que produjeron advertencias y, a continuación, cargue un nuevo almacén de confianza en Amazon S3. Después de cargar el nuevo almacén de confianza, actualice el nombre de dominio personalizado para que utilice ese nuevo almacén de confianza.

Solución de problemas por conflictos de nombres de dominio

El error `"The certificate subject <certSubject> conflicts with an existing certificate from a different issuer."` significa que varias entidades de certificación han emitido un certificado para este dominio. Para cada asunto del certificado, solo puede haber un

emisor en API Gateway para dominios de TLS mutua. Tendrá que obtener todos los certificados para ese asunto a través de un solo emisor. Si el problema se debe a un certificado que no está bajo su control pero puede demostrar la propiedad del nombre de dominio, contacte con Soporte para abrir un ticket.

Solución de problemas por mensajes de estado de nombres de dominio

`PENDING_CERTIFICATE_REIMPORT`: esto significa que ha vuelto a importar un certificado en ACM y ha fallado la validación porque el nuevo certificado tiene un SAN (nombre alternativo de sujeto) que no está cubierto por el `ownershipVerificationCertificate`, o bien porque el asunto o los SAN del certificado no cubren el nombre de dominio. Es posible que haya algo que esté configurado incorrectamente o que se haya importado un certificado no válido. Debe volver a importar un certificado válido en ACM. Para obtener más información sobre la validación, consulte Validación de la propiedad de dominios.

`PENDING_OWNERSHIP_VERIFICATION`: esto significa que el certificado verificado previamente ha caducado y ACM no lo ha podido renovar automáticamente. Tendrá que renovar el certificado o solicitar otro nuevo. Dispone de más información sobre la renovación de certificados en la guía Solución de problemas de renovación de certificados administrados de ACM.

Resolución del problema de certificado devuelto incorrecto

Al migrar un certificado dedicado de un nombre de dominio completo (FQDN) a un nombre de dominio de cliente comodín, API Gateway puede devolver el certificado del FQDN en lugar del nombre de dominio comodín.

El siguiente comando muestra qué certificado devuelve API Gateway:

```
openssl s_client -connect hostname:port
```

Si el certificado resultante es para el FQDN, póngase en contacto con Soporte para abrir un ticket.

Presentación de certificados de cliente a los servicios de backend con TLS mutuo en API Gateway

API Gateway puede presentar un certificado de cliente al backend durante el establecimiento de comunicación de TLS. Luego, el backend puede verificar que las solicitudes provienen de API Gateway. Esto crea una conexión TLS mutua entre API Gateway y el servicio de backend.

Puede elegir entre dos enfoques:

- Certificados (ACM) de AWS Gestor de certificados: importe su propio certificado firmado por una autoridad de certificación desde ACM y adjúntelo a la fase de la API. Con este enfoque, obtiene cadenas de certificados completas, renovación automática e integración con la infraestructura de PKI existente.
- Certificados generados por API Gateway: genere un certificado autofirmado en API Gateway. Este enfoque no requiere una infraestructura de certificados externa. El certificado caduca a los 365 días y debe rotarse manualmente.

Temas

- Cómo funciona la autenticación de certificados de clientes de backend
- Uso del certificado de ACM propio para el TLS mutuo de backend en API Gateway
- Uso de un certificado generado por API Gateway para la autenticación de backend en API Gateway
- Entidades de certificación compatibles con API Gateway para las integraciones HTTP y Proxy HTTP en API Gateway

Cómo funciona la autenticación de certificados de clientes de backend

Con el TLS mutuo saliente (mTLS), API Gateway actúa como un cliente de TLS. Presenta un certificado de cliente al servidor de backend cuando se conecta. El backend comprueba este certificado para confirmar que las solicitudes provienen de API Gateway.

El mTLS saliente es diferente del mTLS entrante. El mTLS entrante requiere que los clientes de la API presenten certificados a API Gateway. El mTLS saliente protege el enlace entre API Gateway y el backend.

Secuencia del establecimiento de comunicación de TLS

Los siguientes pasos muestran cómo API Gateway configura una conexión TLS mutua con el backend:

1. API Gateway envía un mensaje de ClientHello al backend.
2. El backend envía su certificado de servidor a API Gateway.
3. API Gateway comprueba el certificado del servidor.
4. El backend envía una CertificateRequest a API Gateway.

5. API Gateway envía la cadena de certificado de cliente al backend.
6. El backend compara el certificado del cliente con su almacén de confianza.
7. La conexión TLS mutua está lista. API Gateway reenvía la solicitud de API.

Si el backend no solicita un certificado de cliente en el paso 4, API Gateway finaliza el establecimiento de comunicación de TLS sin enviar ninguno. La conexión utiliza el TLS unidireccional estándar.

Comparación de las opciones de certificado

En la siguiente tabla, se comparan los dos enfoques para configurar los certificados de cliente de backend.

Certificados administrados por ACM en comparación con los generados por API Gateway

Característica	Certificados administrados por ACM	Certificados generados por API Gateway
Origen del certificado	El PKI o AWS Autoridad de certificación privada	Autofirmado de API Gateway
Configuración	Haga referencia al certificado en la etapa por su ARN de ACM; administre el ciclo de vida del certificado con las API de ACM.	Adminístrelo en la página de certificados de cliente o con las API de certificados de cliente de API Gateway.
Renovación	Depende de cómo se emitió el certificado. En el caso de los certificados que ACM emite a través de AWS Autoridad de certificación privada, ACM los renueva automáticamente y API Gateway propaga la actualización sin tener que volver a implement arlos ni sufrir tiempos de inactivid ad. En el caso de los certifica dos que importe a ACM, debe volver a importarlos antes de que	Manual. El certificado caduca a los 365 días y debe rotarse manualmente.

Característica	Certificados administrados por ACM	Certificados generados por API Gateway
	caduquen; a continuación, API Gateway propaga la reimportación automáticamente.	
Compatibilidad con cadenas	Cadena completa (hasta 5 certificados)	Un certificado autofirmado
Modelo de confianza del backend	El backend confía en la autoridad de certificación	El backend debe fijar el certificado de API Gateway

Uso del certificado de ACM propio para el TLS mutuo de backend en API Gateway

Puede configurar API Gateway para que presente su propio certificado firmado por una autoridad de certificación a los servicios de backend. Importe su certificado a AWS Gestor de certificados (ACM) o emita uno a través de AWS Autoridad de certificación privada. A continuación, vincule el ARN del certificado de ACM a su etapa de API.

Requisitos previos

Antes de configurar un certificado de cliente de ACM, debe tener lo siguiente:

- Acceso a AWS Gestor de certificados en la misma región que la API.
- Permisos de IAM: `acm:ImportCertificate` y `acm:DescribeCertificate` (para importar con la opción A), o `acm:RequestCertificate` (para la opción B) o `acm-pca:IssueCertificate`, `acm-pca:GetCertificate`, y `acm:ImportCertificate` (para la opción C).
- Una API de REST implementada en una etapa.

Paso 1: Importar un certificado o emitir uno mediante AWS Autoridad de certificación privada

Puede importar un certificado del PKI existente o emitir uno nuevo a través de AWS Autoridad de certificación privada. Ambas rutas producen un ARN de certificado de ACM que se utiliza en el siguiente paso.

> **Note**
>
> Los certificados públicos de ACM no se admiten para la autenticación de clientes de backend. A partir del 11 de junio de 2025, AWS Gestor de certificados ya no emitirá certificados públicos con el uso extendido de claves (EKU) `clientAuth`. Esta característica requiere ese EKU, así que utilice un certificado que importe a ACM o uno emitido a través de AWS Autoridad de certificación privada.

> **Note**
>
> Cree el certificado de ACM en la misma región de AWS que la API de REST que lo utilizará. Los certificados de ACM son recursos regionales, por lo que el certificado debe existir en la región de la API.

Opción A: Importar desde el PKI existente

Para importar un certificado de cliente y su clave privada a ACM, ejecute el siguiente comando. Para obtener más información, consulte Importación de certificados en la Guía del usuario de AWS Gestor de certificados.

```
aws acm import-certificate \
  --certificate fileb://client-cert.pem \
  --private-key fileb://private-key.pem \
  --certificate-chain fileb://ca-chain.pem \
  --region region
```

El comando devuelve el ARN del certificado de ACM. Registre este valor para el siguiente paso.

Opción B: solicitar un certificado a través de AWS Autoridad de certificación privada (administrado por ACM)

Para solicitar un certificado privado que ACM administre y pueda renovar automáticamente, ejecute el siguiente comando. Para obtener más información, consulte Solicitud de un certificado privado en la Guía del usuario de AWS Gestor de certificados.

```
aws acm request-certificate \
```

```
  --domain-name www.example.com \
  --certificate-authority-arn arn:aws:acm-pca:us-east-1:123456789012:certificate-
authority/12345678-1234-1234-1234-123456789012 \
  --region region
```

El comando devuelve el ARN del certificado de ACM. Registre este valor para el siguiente paso.

Opción C: emitir a través de AWS Autoridad de certificación privada e importar a ACM

Si necesita controlar directamente los parámetros del certificado (como las extensiones personalizadas o los algoritmos de firma), puede emitir un certificado a través de AWS Autoridad de certificación privada e importarlo a ACM. ACM no renueva automáticamente los certificados importados de esta manera. Asegúrese de que el certificado cumpla con los requisitos del certificado. Para obtener más información sobre la emisión de certificados privados, consulte Emitir un certificado de entidad final privado en la Guía del usuario de AWS Autoridad de certificación privada.

```
aws acm-pca issue-certificate \
  --certificate-authority-arn arn:aws:acm-pca:us-east-1:123456789012:certificate-
authority/12345678-1234-1234-1234-123456789012 \
  --csr fileb://csr.pem \
  --signing-algorithm SHA256WITHRSA \
  --validity Value=365,Type=DAYS
```

> **Recuperación e importación del certificado**
>
> El comando `issue-certificate` devuelve un ARN de certificado AWS Autoridad de certificación privada, no un ARN de ACM. Para usar este certificado con API Gateway, recupérelo con `aws acm-pca get-certificate` y, a continuación, impórtelo a ACM mediante `aws acm import-certificate`. La importación produce un ARN de certificado de ACM que se utiliza en el siguiente paso. Cuando ejecute `aws acm import-certificate`, establezca `--region` en la región de la API para que el certificado de ACM se cree allí.

Paso 2: Configurar una etapa de API para usar el certificado de ACM

Una vez que tenga un ARN de certificado de ACM, configure la etapa de API para presentar el certificado al backend.

Configuración de una etapa (consola)

1. Abra la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija la API de REST.
3. Elija Etapas.
4. En la sección Detalles de la etapa, elija Editar.
5. En Certificado de cliente, seleccione el certificado de ACM en la lista desplegable.
6. Seleccione Save changes (Guardar cambios).

Configuración de una etapa (AWS CLI)

Use el siguiente comando:

```
aws apigateway update-stage \
  --rest-api-id abc123 \
  --stage-name prod \
  --patch-operations op='replace',path=/clientCertificateId,value=arn:aws:acm:us-
east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
```

> **Note**
>
> API Gateway usa el mismo campo `clientCertificateId` para los certificados generados por ACM y API Gateway. Al proporcionar un ARN de certificado de ACM, API Gateway detecta automáticamente el formato y utiliza el flujo de trabajo administrado por ACM.

Paso 3: Comprobar la configuración

Para verificar que API Gateway envía el certificado al backend, complete los siguientes pasos:

> **El backend debe solicitar un certificado de cliente**
>
> El backend debe estar configurado para solicitar un certificado de cliente durante el establecimiento de comunicación de TLS. Si el backend no solicita uno, API Gateway no presenta el certificado.

1. Invoque el punto de conexión de la API.

2. Compruebe que el backend obtiene el certificado de cliente durante el establecimiento de comunicación de TLS.
3. Compruebe que el backend acepta el certificado y devuelve una respuesta correcta.

Si el backend rechaza el certificado, compruebe que la cadena de certificados se pueda validar con el almacén de confianza del backend.

Requisitos del certificado

El certificado de hoja que configure debe cumplir los siguientes requisitos.

Requisitos del certificado de cliente de ACM

Requisito	Descripción
Longitud máxima de la cadena	5 certificados
Validez	El certificado no debe estar caducado ni pendiente de validez cuando lo configure
Región	El certificado de ACM debe estar en la misma región que la API
Cuenta	El certificado de ACM debe estar en la misma cuenta que la API
Uso extendido de claves (EKU)	Si está presente, debe incluir `clientAuth` . Si no existe, el certificado se acepta.
Uso de claves (KU)	Si está presente, debe incluir `digitalSignature` o `keyAgreement` . Si no existe, el certificado se acepta.
Algoritmo clave	Debe ser uno de los siguientes: RSA 2048, RSA 3072, RSA 4096, ECDSA P-256 (EC_prime256v1), ECDSA P-384 (EC_secp384r1) o ECDSA P-521 (EC_secp521r1)
Estado del certificado de ACM	Debe ser `ISSUED`

> **Note**
>
> API Gateway no valida la cadena de confianza entre el certificado hoja y el intermedio. API Gateway tampoco valida la intención del certificado ni las restricciones básicas (por ejemplo,

`CA:TRUE`) de los certificados intermedios. El backend realiza estas validaciones durante el establecimiento de comunicación de TLS.

Renovación y propagación de certificados

Cuando un certificado cambia en ACM, API Gateway detecta la actualización y propaga el nuevo certificado automáticamente. No necesita volver a implementar la etapa y la API no sufre ningún tiempo de inactividad durante la rotación.

La propagación de certificados es de coherencia final. Durante la actualización, es posible que el backend reciba el certificado anterior o el nuevo hasta que se complete la propagación.

La forma en que se renueva el certificado depende de cómo se haya emitido:

- Certificados emitidos mediante AWS Autoridad de certificación privada (administrados por ACM) (opción B): ACM renueva automáticamente estos certificados. API Gateway detecta la renovación y se actualiza automáticamente.
- Certificados emitidos por AWS Autoridad de certificación privada e importados (opción C): ACM no renueva automáticamente los certificados importados. Debe volver a importar el certificado renovado. Tras volver a importar el certificado, API Gateway detecta el cambio y lo actualiza automáticamente.
- Certificados importados del PKI (opción A): debe volver a importar el certificado renovado a ACM. Tras volver a importar el certificado, API Gateway detecta el cambio y lo actualiza automáticamente.

ACM envía notificaciones de caducidad de certificados a través de Amazon EventBridge. Puede utilizar estas notificaciones para configurar alarmas antes de que caduque un certificado.

Comportamiento y limitaciones del certificado de ACM

Visualización del certificado configurado

Los certificados de ACM no aparecen en las respuestas de la API `GetClientCertificate` ni `GetClientCertificates`. Para ver el ARN del certificado de ACM configurado en una etapa, utilice GetStage. Para ver los detalles del certificado, utilice las API de ACM DescribeCertificate y GetCertificate.

Reutilización entre etapas

Puede adjuntar el mismo certificado de ACM a varias etapas. Cada etapa hace referencia al certificado de forma independiente por su ARN.

Las API de certificados de cliente no se aplican a los certificados de ACM

Los certificados de ACM no son recursos administrados por API Gateway. Las API `GetClientCertificate`, `UpdateClientCertificate` y `DeleteClientCertificate` devuelven una `NotFoundException` cuando se llaman con un ARN de certificado de ACM. Use las API de ACM para administrar el ciclo de vida del certificado.

Limpieza automática de asociaciones de certificados

Cuando elimina un certificado de ACM de una etapa, actualice una etapa para usar un certificado diferente o elimine una etapa o una API de REST, API Gateway limpia la asociación de certificados automáticamente. No se requiere ninguna acción manual.

Eliminación del certificado de ACM

ACM no permite eliminar un certificado mientras API Gateway tenga una asociación activa con él. Para eliminar el certificado de ACM, elimínelo primero de todas las etapas que hacen referencia a él.

Uso de un certificado generado por API Gateway para la autenticación de backend en API Gateway

Puede utilizar API Gateway para generar un certificado SSL y usar su clave pública en el backend para verificar que las solicitudes HTTP a su sistema backend proceden de API Gateway. Esto permite que el backend HTTP controle y acepte únicamente las solicitudes procedentes de Amazon API Gateway, incluso si el backend está disponible públicamente.

> **Note**
>
> Puesto que algunos servidores backend no son compatibles con la autenticación SSL de los clientes de la manera en que lo es API Gateway, podrían devolver un error de certificado SSL. Para obtener una lista de servidores del backend incompatibles, consulte the section called “Notas importantes”.

> **Tip**
>
> Para usar su propio certificado firmado por una autoridad de certificación de ACM en lugar de un certificado autofirmado generado por API Gateway, consulte the section called “Uso del certificado de ACM propio”.

Los certificados SSL generados por API Gateway son autofirmados y solo la clave pública de un certificado está visible en la consola de API Gateway o a través de las API.

Temas

- Generar un certificado cliente mediante la consola de API Gateway
- Configurar una API para que use certificados SSL
- Test Invoke para verificar la configuración del certificado cliente
- Configuración del servidor HTTPS backend para verificar el certificado cliente
- Rotar un certificado cliente que va a caducar

Generar un certificado cliente mediante la consola de API Gateway

1. Abra la consola de Amazon API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API de Rest o de WebSocket.
3. En el panel de navegación principal, elija Certificados de cliente.
4. En el panel Certificados de cliente, elija Generar certificado.
5. (Opcional) En Description (Descripción), introduzca una descripción.
6. Elija Generar certificado para generar el certificado. API Gateway genera un certificado nuevo y devuelve el GUID del nuevo certificado junto con la clave pública codificada en PEM.

Ahora está listo para configurar una API para que utilice el certificado.

Configurar una API para que use certificados SSL

En estas instrucciones, se presupone que usted ya ha completado Generar un certificado cliente mediante la consola de API Gateway.

1. En la consola de API Gateway, cree o abra una API de REST o de WebSocket para la que desee utilizar el certificado de cliente. Asegúrese de que la API se ha implementado en una etapa.
2. En el panel de navegación principal, elija Etapas.
3. En la sección Detalles de la etapa, elija Editar.
4. En el Certificado de cliente, seleccione un certificado.
5. Seleccione Save changes (Guardar cambios).

Una vez seleccionado y guardado un certificado para la API, API Gateway utilizará el certificado para todas las llamadas a las integraciones HTTP de la API.

Test Invoke para verificar la configuración del certificado cliente

1. Elija un método de la API de REST. Elija la pestaña Prueba. Puede que tenga que elegir el botón de flecha hacia la derecha para mostrar la pestaña Pruebas.
2. En el Certificado de cliente, seleccione un certificado.
3. Seleccione Probar

API Gateway presentará el certificado SSL elegido al backend HTTP para autenticar la API.

Configuración del servidor HTTPS backend para verificar el certificado cliente

En estas instrucciones, se presupone que ya ha completado Generar un certificado cliente mediante la consola de API Gateway y ha descargado una copia del certificado cliente. Puede descargar un certificado de cliente mediante una llamada a `clientcertificate:by-id` de la API REST de API Gateway o a `get-client-certificate` de la AWS CLI.

Antes de configurar un servidor HTTPS backend para verificar el certificado SSL cliente de API Gateway, debe haber obtenido la clave privada codificada en PEM y un certificado de servidor proporcionado por una entidad de certificación de confianza.

Si el nombre de dominio del servidor es `myserver.mydomain.com`, el valor de CNAME del certificado del servidor debe ser `myserver.mydomain.com` o `*.mydomain.com`.

Las entidades de certificación admitidas incluyen Let's Encrypt o alguna de las the section called “Entidades de certificación compatibles para las integraciones HTTP y Proxy HTTP”.

Como ejemplo, supongamos que el archivo de certificado de cliente es `apig-cert.pem` y los archivos de clave privada del servidor y certificado son `server-key.pem` y `server-cert.pem`, respectivamente. Para un servidor Node.js en el backend, puede configurar el servidor de manera similar a la siguiente:

```
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);
```

Para una aplicación node-express, puede usar los módulos client-certificate-auth para autenticar las solicitudes cliente con certificados codificados en PEM.

Para el resto de servidores HTTPS, consulte la documentación del servidor.

Rotar un certificado cliente que va a caducar

El certificado cliente generado por API Gateway es válido durante 365 días. Debe rotar el certificado antes de que caduque un certificado cliente en una etapa de API para evitar tiempos de inactividad de la API.

Rotación de un certificado cliente que va a caducar mediante la Consola de administración de AWS

El siguiente procedimiento muestra cómo rotar un certificado de cliente en la consola para una API implementada anteriormente.

1. En el panel de navegación principal, elija Certificados de cliente.
2. En el panel Certificados de cliente, elija Generar certificado.
3. Abra la API para la que desee utilizar el certificado cliente.
4. Elija Stages (Etapas) en la API seleccionada y, a continuación, elija una etapa.
5. En la sección Detalles de la etapa, elija Editar.

6. En Certificado de cliente, seleccione el nuevo certificado.
7. Para guardar la configuración, elija Guardar cambios.

Rotación de un certificado de cliente que va a caducar mediante la AWS CLI

Puede verificar la fecha de vencimiento del certificado mediante una llamada a clientCertificate:by-id de la API de REST de API Gateway o al comando AWS CLI de get-client-certificate y mediante la inspección de la propiedad expirationDate devuelta.

Para rotar un certificado de cliente, haga lo siguiente:

1. Genere un nuevo certificado de cliente mediante una llamada a clientcertificate:generate de la API REST de API Gateway o al comando de la AWS CLI generate-client-certificate. En este tutorial, suponemos que el nuevo ID del certificado cliente es `ndiqef`.
2. Actualice el servidor backend para incluir el nuevo certificado cliente. No elimine el certificado cliente existente todavía.

   Algunos servidores pueden requerir reiniciar para finalizar la actualización. Consulte la documentación del servidor para saber si debe reiniciar el servidor durante la actualización.
3. Actualice la etapa de la API de forma que utilice el nuevo certificado cliente llamando a stage:update de la API REST de API Gateway, con el nuevo ID de certificado cliente (`ndiqef`):

   ```
   PATCH /restapis/{restapi-id}/stages/stage1 HTTP/1.1
   Content-Type: application/json
   Host: apigateway.us-east-1.amazonaws.com
   X-Amz-Date: 20170603T200400Z
   Authorization: AWS4-HMAC-SHA256 Credential=...

   {
     "patchOperations" : [
       {
           "op" : "replace",
           "path" : "/clientCertificateId",
           "value" : "ndiqef"
       }
     ]
   }
   ```

   También puede usar el comando update-stage.

Si está utilizando una API de WebSocket, use el comando update-stage de `apigatewayv2`.

4. Actualice el servidor backend para eliminar el certificado antiguo.
5. Elimine el certificado antiguo de API Gateway llamando al comando clientcerfiticate:delete de la API REST de API Gateway y especificando el clientCertificateId (`a1b2c3`) del certificado antiguo:

```
DELETE /clientcertificates/a1b2c3
```

También puede llamar al comando delete-client-certificate:

```
aws apigateway delete-client-certificate --client-certificate-id a1b2c3
```

Entidades de certificación compatibles con API Gateway para las integraciones HTTP y Proxy HTTP en API Gateway

En la siguiente lista se incluyen las entidades de certificación que admite API Gateway para las integraciones HTTP, proxy HTTP y privadas.

```
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
```

Uso de AWS WAF para proteger sus API de REST en API Gateway

AWS WAF es un firewall de aplicaciones web que ayuda a proteger las aplicaciones web y las API de ataques. Le permite configurar un conjunto de reglas denominadas lista de control de acceso web (Web ACL) que permiten, bloquean o cuentan solicitudes web en función de las reglas y condiciones de seguridad web personalizables que defina. Para obtener más información, consulte Funcionamiento de AWS WAF.

Puede utilizar AWS WAF para proteger a la API de REST de API Gateway de vulnerabilidades web comunes, como ataques de inyección de código SQL y scripting entre sitios (XSS). Esto

podría afectar a la disponibilidad y el rendimiento de la API, comprometer la seguridad o consumir recursos excesivos. Por ejemplo, puede crear reglas para permitir o bloquear solicitudes de rangos de direcciones IP especificados, solicitudes de bloques CIDR, solicitudes que se originan en un país o región específico, solicitudes que contengan código SQL malintencionado o solicitudes que contengan secuencias de comandos malintencionadas.

También puede crear reglas que busquen una cadena o un patrón de expresión regular en encabezados HTTP, métodos, cadenas de consulta, URI y el cuerpo de la solicitud (limitado a los primeros 64 KB). Además, puede crear reglas para bloquear ataques de agentes de usuario específicos, bots malintencionados y scrapers de contenido. Por ejemplo, puede utilizar reglas basadas en la frecuencia para especificar el número de solicitudes web permitidas por IP de cliente en un periodo de 5 minutos actualizado constantemente.

> **Important**
>
> AWS WAF es su primera línea de defensa contra vulnerabilidades de la web. Cuando AWS WAF está habilitado en una API, se evalúan las reglas de AWS WAF antes que otras características del control de acceso, como, por ejemplo, las políticas de recursos, las políticas de IAM, los autorizadores de Lambda y los autorizadores de Amazon Cognito. Por ejemplo, si AWS WAF bloquea el acceso de un bloque de CIDR permitido por una política de recursos, AWS WAF tiene prioridad y no se evalúa la política de recursos.

Para habilitar AWS WAF para su API, debe hacer lo siguiente:

1. Utilice la consola de AWS WAF, el AWS SDK o la CLI para crear una ACL web que contenga la combinación deseada de reglas administradas de AWS WAF y sus propias reglas personalizadas. Para obtener más información, consulte Getting Started with AWS WAF y Web access control lists (web ACLs).

   > **Important**
   >
   > API Gateway requiere una ACL web de AWS WAFV2 para una aplicación regional o una ACL web de AWS WAF Classic Regional.

2. Asociar la ACL web de AWS WAF a una etapa de API. Puede hacerlo utilizando la consola de AWS WAF, el SDK de AWS, la CLI o la consola de API Gateway.

Asociación de una ACL web de AWS WAF a una etapa de la API de API Gateway mediante la consola de API Gateway

Si desea utilizar la consola de API Gateway para asociar una ACL web de AWS WAF a una etapa de la API de API Gateway existente, siga estos pasos:

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Seleccione una API existente o cree una nueva.
3. En el panel de navegación principal, elija Etapas y, a continuación, elija una etapa.
4. En la sección Detalles de la etapa, elija Editar.
5. En Web Application Firewall (AWS WAF), seleccione la ACL web.

   Si utiliza AWS WAFV2, seleccione una ACL web de AWS WAFV2 para una aplicación regional. La ACL web y todos los recursos de AWS WAFV2 que utilice deben estar ubicados en la misma región que la API.

   Si utiliza AWS WAF Classic Regional, seleccione una ACL web regional.
6. Seleccione Save changes (Guardar cambios).

Asociación de una ACL web de AWS WAF a una etapa de la API de API Gateway mediante la AWS CLI

El siguiente comando associate-web-acl permite asociar una ACL web de AWS WAFV2 de una aplicación regional a una etapa de API de API Gateway existente:

```
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
```

El siguiente comando associate-web-acl permite asociar una ACL web de AWS WAF Classic Regional a una etapa de API de API Gateway existente:

```
aws waf-regional associate-web-acl \
--web-acl-id 'aabc123a-fb4f-4fc6-becb-2b00831cadcf' \
--resource-arn 'arn:aws:apigateway:{region}::/restapis/4wk1k4onj3/stages/prod'
```

Asociación de una ACL web de AWS WAF a una etapa de API mediante la API de REST de AWS WAF

Si desea utilizar la API de REST de AWS WAFV2 para asociar una ACL web de AWS WAFV2 para una aplicación regional a una etapa de la API de API Gateway existente, use el comando AssociateWebACL, como se muestra en el siguiente ejemplo:

```
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'
)
```

Si desea utilizar la API de REST de AWS WAF para asociar una ACL web de AWS WAF Classic Regional a una etapa de la API de API Gateway existente, use el comando AssociateWebACL, como se muestra en el siguiente ejemplo:

```
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'
)
```

Limitación de las solicitudes a sus API de REST para mejorar el rendimiento en API Gateway

Puede configurar la limitación y las cuotas de las API para evitar que se vean desbordadas por un número excesivo de solicitudes. Tanto los límites como las cuotas se aplican en la medida de lo posible y deben considerarse como objetivos en lugar de como límites de solicitud garantizados.

API Gateway limita las solicitudes a la API utilizando el algoritmo de bucket de tokens, donde un token da cuenta de una solicitud. En concreto, API Gateway examina el ratio de solicitudes y una

ráfaga de envíos de solicitudes para todas las API de su cuenta, por región. En el algoritmo de bucket de tokens, una ráfaga puede permitir que se sobrepasen los límites predefinidos, pero también hay otros factores que pueden hacer que se sobrepasen los límites en algunos casos.

Cuando los envíos de solicitudes superan los límites de ratio de solicitudes en estado estable y de ráfaga, API Gateway comienza a limitar las solicitudes. Los clientes pueden recibir respuestas de error `429 Too Many Requests` en este momento. Tras capturar estas excepciones, el cliente puede reenviar las solicitudes que han producido un error de forma que limite el ratio.

Como desarrollador de la API, puede configurar los límites para distintas etapas o métodos de la API con el fin de mejorar el rendimiento general de todas las API de su cuenta. También puede activar planes de uso para configurar las limitaciones en los envíos de solicitudes de cliente en función de los ratios y cuotas de solicitudes.

Temas

- Cómo se aplica la configuración de limitación controlada en API Gateway
- Limitación de nivel de cuenta por región
- Configuración de objetivos de la limitación controlada de nivel de API y de nivel de etapa en un plan de uso
- Configuración de objetivos de limitación de nivel de etapa
- Configuración de objetivos de limitación controlada a nivel de método en un plan de uso

Cómo se aplica la configuración de limitación controlada en API Gateway

Antes de establecer la configuración de limitación y cuota para la API, es recomendable entender de qué manera la configuración de limitación está relacionada con la configuración de limitación para la API y de qué manera la aplica API Gateway.

Amazon API Gateway proporciona de cuatro tipos básicos de opciones relacionadas con la limitación controlada:

- Los límites de limitación de AWS se aplican a todas las cuentas y clientes de una región. Estos límites evitan que la API, y su cuenta, resulten colapsadas por un enorme número de solicitudes. AWS establece estos límites y el cliente no puede modificarlos.
- Los límites por cuenta se aplican a todas las API de una cuenta en una región determinada. El límite de ratio de nivel de cuenta se puede aumentar previa solicitud. Los límites más altos son posibles con API que tienen tiempos de espera más cortos y cargas útiles más pequeñas. Para

solicitar un aumento de los límites de limitación de nivel de cuenta por región, contacte con el Centro de soporte de AWS. Para obtener más información, consulte Cuotas y notas importantes. Tenga en cuenta que estos límites no pueden ser superiores a los límites de limitación controlada de AWS.

- Los límites de limitación controlada por API y por fase se aplican a nivel de método API para una etapa. Puede configurar los mismos ajustes para todos los métodos o configurar diferentes ajustes de limitación controlada para cada método. Tenga en cuenta que estos límites no pueden ser superiores a los límites de limitación controlada de AWS.
- Las limitaciones controladas por cliente se aplican a los clientes que utilizan claves de API asociadas a su plan de uso como identificador del cliente. Tenga en cuenta que estos límites no pueden ser superiores a los límites por cuenta.

API Gateway aplica la configuración relacionada con la limitación en el orden siguiente:

1. Limitación controlada por método o por cliente establecida para una etapa de API en un plan de uso
2. Limitaciones por método que establece para una etapa de la API
3. Limitación de nivel de cuenta por región
4. AWSLimitación controlada regional de

Limitación de nivel de cuenta por región

De forma predeterminada, API Gateway limita las solicitudes de estado constante por segundo (RPS) en todas las API de una cuenta de AWS, por región. También limita la ráfaga (es decir, el tamaño máximo del bucket) en todas las API dentro de una cuenta de AWS, por región. En API Gateway, el límite de ráfaga representa el número máximo de envíos de solicitudes simultáneas que API Gateway; abordará antes de devolver respuestas de error `429 Too Many Requests`. Para obtener más información sobre las cuotas de limitación, consulte Cuotas y notas importantes.

Configuración de objetivos de la limitación controlada de nivel de API y de nivel de etapa en un plan de uso

En un plan de uso, puede definir un objetivo de limitación controlada de método predeterminado para todos los métodos en el nivel de API o de etapa. Puede especificar una tasa de limitación, que es la velocidad, en solicitudes por segundo, a la que se agregan los tokens al bucket de tokens. También puede especificar una ráfaga de limitación, que es la capacidad del bucket de tokens.

Puede usar la Consola de administración de AWS, la CLI y los SDK de AWS para crear un plan de uso. Para obtener más información sobre cómo crear un plan de uso, consulte ???.

Configuración de objetivos de limitación de nivel de etapa

Puede utilizar la CLI de AWS, los SDK y la Consola de administración de AWS para crear objetivos de limitación a nivel de etapa.

Para obtener más información sobre cómo utilizar la Consola de administración de AWS para crear objetivos de limitación en el nivel de etapa, consulte ???. Para obtener más información sobre cómo utilizar la CLI de AWS para crear objetivos de limitación en el nivel de etapa, consulte create-stage.

Configuración de objetivos de limitación controlada a nivel de método en un plan de uso

Puede definir objetivos adicionales de la limitación controlada a nivel de método en Usage Plans (Planes de uso) como se muestra en Crear un plan de uso. Para establecerlos en la consola de API Gateway, hay que especificar `Resource=<resource>`, `Method=<method>` en Configure Method Throttling (Configurar limitación controlada de método). Por ejemplo, en el ejemplo de PetStore, puede especificar `Resource=/pets`, `Method=GET`.

API de REST privadas en API Gateway

Una API privada es una API de REST a la que solo se puede llamar desde dentro de una Amazon VPC. Puede acceder a la API con un punto de conexión de VPC de interfaz, que es una interfaz de red de punto de conexión que crea en la VPC. Los puntos de conexión de la interfaz tienen tecnología AWS PrivateLink, que le permite obtener acceso de forma privada a los servicios de AWS mediante direcciones IP privadas.

También puede utilizar Direct Connect para establecer una conexión desde una red en las instalaciones a Amazon VPC y, a continuación, acceder a la API privada a través de esa conexión. En cualquier caso, el tráfico dirigido a la API privada utiliza conexiones seguras y está aislado de la Internet pública. El tráfico no sale de la red de Amazon.

Prácticas recomendadas para API privadas

Recomendamos que utilice las siguientes prácticas recomendadas al crear la API privada:

- Utilice un punto de conexión de VPC único para acceder a varias API privadas. De este modo, se reduce el número de puntos de conexión de VPC que es posible que necesite.

- Asocie el punto de conexión de VPC a la API. De este modo, se crea un registro de DNS de alias de Route 53 y se simplifica la invocación a la API privada.
- Active el DNS privado de la VPC. Cuando active el DNS privado para la VPC, puede invocar la API dentro de una VPC sin tener que pasar el encabezado `Host` o `x-apigw-api-id`.

  Si habilita el DNS privado, no podrá acceder al punto de conexión predeterminado para las API públicas. Para acceder al punto de conexión predeterminado de las API públicas, puede desactivar el DNS privado, crear una zona alojada privada para cada API privada de la VPC y, a continuación, aprovisionar los registros necesarios en Route 53. Esto permite que la API privada se resuelva mientras puede invocar el punto de conexión público predeterminado desde la VPC. Para obtener información, consulte Crear una zona alojada privada.
- Restrinja el acceso a la API privada a VPC o puntos de conexión de VPC específicos. Agregue las condiciones `aws:SourceVpc` o `aws:SourceVpce` a la política de recursos de la API para restringir el acceso.
- Para lograr el perímetro de datos más seguro, puede crear una política de punto de conexión de VPC. Esto controla el acceso a los puntos de conexión de VPC que pueden invocar la API privada.

Consideraciones sobre API privadas

Las siguientes consideraciones pueden afectar al uso de las API privadas:

- Solo se admiten API de REST.
- Sin embargo, no puede convertir una API privada en una API optimizada para límites.
- Las API privadas solo admiten TLS 1.2. No se admiten versiones de TLS anteriores.
- Si realiza una solicitud utilizando el protocolo HTTP/2, se exigirá que la solicitud use el protocolo HTTP/1.1.
- No puede configurar el tipo de dirección IP de las API privadas para permitir que solo las direcciones IPv4 invoquen la API privada. Solo se admite pila doble. Para obtener más información, consulte the section called “Tipos de direcciones IP para API de REST en API Gateway”.
- Para enviar tráfico con la API privada, puede usar todos los tipos de direcciones IP compatibles con Amazon VPC. Puede enviar tráfico de pila doble e IPv6 configurando los ajustes en el punto de conexión de VPC. No puede modificarlo con API Gateway. Para obtener más información, consulte Adición de la compatibilidad de IPv6 con su VPC.

- Los puntos de conexión de VPC de las API privadas están sujetos a las mismas limitaciones que otros puntos de conexión de VPC de tipo interfaz. Para obtener más información, consulte Acceso a un servicio de AWS a través de un punto de conexión de VPC de interfaz en la Guía de AWS PrivateLink. Para obtener más información sobre cómo utilizar API Gateway con VPC compartidas y subredes compartidas, consulte Subredes compartidas en la Guía de AWS PrivateLink.

Próximos pasos para API privadas

Para obtener información sobre cómo crear una API privada y asociar un punto de conexión de VPC, consulte the section called “Creación de una API privada”. Para seguir un tutorial en el que se crean dependencias en CloudFormation y una API privada en la Consola de administración de AWS, consulte the section called “Tutorial: Creación de una API de REST privada”.

Creación de una API privada

Antes de crear una API privada, debe crear un punto de conexión de VPC para API Gateway. A continuación, debe crear la API privada y adjuntarle una política de recursos. Si lo desea, puede asociar el punto de conexión de VPC a la API privada para simplificar la forma de invocar la API. Por último, debe implementar la API.

En los procedimientos siguientes se describe cómo realizar esto. Puede crear una API de REST privada mediante la Consola de administración de AWS, la AWS CLI o un AWS SDK.

Requisitos previos

Para seguir estos pasos, debe tener una VPC completamente configurada. Para obtener información sobre cómo crear una VPC, consulte Crear solo una VPC en la Guía del usuario de Amazon VPC. Para seguir todos los pasos recomendados al crear la VPC, habilite el DNS privado. De esta forma, puede invocar la API dentro de una VPC sin tener que pasar el host o el encabezado `x-apigw-api-id`.

Para habilitar el DNS privado, los atributos `enableDnsSupport` y `enableDnsHostnames` de la VPC deben estar establecidos en `true`. Para obtener más información, consulte Compatibilidad de DNS en la VPC y Actualización de la compatibilidad de DNS para la VPC.

Paso 1: Creación de un punto de conexión de VPC para API Gateway en la VPC

En el siguiente procedimiento, se muestra cómo crear un punto de conexión de VPC para API Gateway. Para crear un punto de conexión de VPC para API Gateway, debe especificar el dominio

`execute-api` para la Región de AWS donde crea la API privada. El dominio `execute-api` es el componente de servicio de API Gateway para la ejecución de la API.

Cuando crea el punto de conexión de VPC para API Gateway, debe especificar la configuración de DNS. Si desactiva el DNS privado, solo podrá acceder a la API mediante un DNS público. Para obtener más información, consulte the section called “Problema: No puedo conectarme a mi API pública desde un punto de conexión de VPC de API Gateway”.

Consola de administración de AWS

Creación de un punto de conexión de VPC de la interfaz para API Gateway

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc/.
2. En el panel de navegación, en Nube virtual privada, seleccione Puntos de conexión.
3. Seleccione Crear punto de conexión.
4. (Opcional) Para Etiqueta de nombre, ingrese un nombre que ayude a identificar el punto de conexión de VPC.
5. En Service category (Categoría de servicios), elija AWSServices (Servicios de AWC).
6. En Servicios, en la barra de búsqueda, ingrese **`execute-api`**. A continuación, elija el punto de conexión del servicio de API Gateway en la Región de AWS en la que creará la API. El nombre del servicio debe tener el mismo aspecto que `com.amazonaws.us-east-1.execute-api` y el Tipo debe ser Interfaz.
7. Para VPC, elija la VPC en la que desea crear el punto de conexión.
8. (Opcional) Para desactivar la opción Habilitar el nombre de DNS privado, elija Configuración adicional y, a continuación, desactive Habilitar el nombre de DNS privado.
9. En Subredes, elija las zonas de disponibilidad en las que creó las interfaces de red de punto de conexión. Para mejorar la disponibilidad de la API, elija varias subredes.
10. En Security group (Grupo de seguridad), seleccione el grupo de seguridad que se va a asociar con las interfaces de red de punto de conexión de VPC.

    El grupo de seguridad que elija debe configurarse de modo que permita el tráfico HTTPS de entrada en el puerto 443 TCP desde un intervalo de direcciones IP de la VPC o desde otro grupo de seguridad de la VPC.
11. Para Política, realice una de las siguientes operaciones:

- Si no ha creado la API privada o no quiere configurar una política de punto de conexión de VPC personalizada, elija Acceso total.
- Si ya ha creado una API privada y desea configurar una política de punto de conexión de VPC personalizada, puede ingresar una política de punto de conexión de VPC personalizada. Para obtener más información, consulte the section called “Uso de políticas de punto de conexión de VPC para API privadas”.

Puede actualizar la política de punto de conexión de VPC después de crear el punto de conexión de VPC. Para obtener más información, consulte Actualizar una política de punto de conexión de VPC.

12. Seleccione Crear punto de conexión.
13. Copie el ID de punto de conexión de VPC resultante, ya que es posible que lo use en futuros pasos.

AWS CLI

El siguiente comando create-vpc-endpoint permite crear un punto de conexión de VPC:

```
aws ec2 create-vpc-endpoint \
    --vpc-id vpc-1a2b3c4d \
    --vpc-endpoint-type Interface \
    --service-name com.amazonaws.us-east-1.execute-api \
    --subnet-ids subnet-7b16de0c \
    --security-group-id sg-1a2b3c4d
```

Copie el ID de punto de conexión de VPC resultante, ya que es posible que lo use en futuros pasos.

Paso 2: crear una API privada

Después de crear el punto de conexión de VPC, se crea una API de REST privada. El siguiente procedimiento muestra cómo crear una API privada.

Consola de administración de AWS

Creación de una API privada

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.

2. Seleccione Create API (Crear API).
3. En REST API, elija Build (Compilación).
4. En Nombre, ingrese un nombre.
5. (Opcional) En Description (Descripción), introduzca una descripción.
6. En Tipo de punto de conexión de la API, seleccione Privado.
7. (Opcional) Para los ID de punto de conexión de VPC, ingrese un ID de punto de conexión de VPC.

   Si asocia un ID de punto de conexión de VPC a la API privada, puede invocar la API desde dentro de la VPC sin anular un encabezado `Host` ni pasar un `x-apigw-api-id header`. Para obtener más información, consulte the section called “(Opcional) Asociación o desasociación de un punto de conexión de VPC con una API privada”.
8. Para Tipo de dirección IP, elija Dualstack.
9. Seleccione Crear API.

Una vez realizados los pasos anteriores, puede seguir las instrucciones en the section called “Introducción a la consola de la API de REST” para configurar métodos e integraciones para esta API, pero no puede implementar la API. Para implementar la API, siga el paso 3 y adjunte una política de recursos a la API.

AWS CLI

El siguiente comando create-rest-api permite crear una API privada:

```
aws apigateway create-rest-api \
        --name 'Simple PetStore (AWS CLI, Private)' \
        --description 'Simple private PetStore API' \
        --region us-west-2 \
        --endpoint-configuration '{ "types": ["PRIVATE"], "ipAddressType":
 "dualstack" }'
```

Una llamada correcta devuelve un resultado similar al siguiente:

```
{
    "createdDate": "2017-10-13T18:41:39Z",
    "description": "Simple private PetStore API",
    "endpointConfiguration": {
        "types": [
```

```
            "PRIVATE"
        ],
        "ipAddressType": "dualstack"
    },
    "id": "0qzs2sy7bh",
    "name": "Simple PetStore (AWS CLI, Private)"
}
```

Una vez realizados los pasos anteriores, puede seguir las instrucciones en the section called "Tutorial: Creación de una API de REST mediante SDK de AWS o la AWS CLI" para configurar métodos e integraciones para esta API, pero no puede implementar la API. Para implementar la API, siga el paso 3 y adjunte una política de recursos a la API.

SDK JavaScript v3

El siguiente ejemplo muestra cómo crear una API privada mediante AWS SDK para JavaScript v3:

```
import {APIGatewayClient, CreateRestApiCommand} from "@aws-sdk/client-api-gateway";
const apig = new APIGatewayClient({region:"us-east-1"});

const input = { // CreateRestApiRequest
  name: "Simple PetStore (JavaScript v3 SDK, private)", // required
  description: "Demo private API created using the AWS SDK for JavaScript v3",
  version: "0.00.001",
  endpointConfiguration: { // EndpointConfiguration
    types: [ "PRIVATE"],
  },
};

export const handler = async (event) => {
const command = new CreateRestApiCommand(input);
try {
  const result = await apig.send(command);
  console.log(result);
} catch (err){
  console.error(err)
 }
};
```

Una llamada correcta devuelve un resultado similar al siguiente:

```
{
```

```
  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'
}
```

Una vez realizados los pasos anteriores, puede seguir las instrucciones en the section called "Tutorial: Creación de una API de REST mediante SDK de AWS o la AWS CLI" para configurar métodos e integraciones para esta API, pero no puede implementar la API. Para implementar la API, siga el paso 3 y adjunte una política de recursos a la API.

Python SDK

El siguiente ejemplo muestra cómo crear una API privada mediante AWS SDK para Python:

```
import json
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

def lambda_handler(event, context):
    try:
      result = apig.create_rest_api(
      name='Simple PetStore (Python SDK, private)',
      description='Demo private API created using the AWS SDK for Python',
      version='0.00.001',
      endpointConfiguration={
          'types': [
             'PRIVATE',
          ],
      },
      )
    except botocore.exceptions.ClientError as error:
            logger.exception("Couldn't create private API %s.", error)
            raise
    attribute=["id", "name", "description", "createdDate", "version",
 "apiKeySource", "endpointConfiguration"]
```

```
    filtered_data ={key:result[key] for key in attribute}
    result = json.dumps(filtered_data, default=str, sort_keys='true')
    return result
```

Una llamada correcta devuelve un resultado similar al siguiente:

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

Una vez realizados los pasos anteriores, puede seguir las instrucciones en the section called "Tutorial: Creación de una API de REST mediante SDK de AWS o la AWS CLI" para configurar métodos e integraciones para esta API, pero no puede implementar la API. Para implementar la API, siga el paso 3 y adjunte una política de recursos a la API.

Paso 3: Configuración de una política de recursos para una API privada

La API privada actual es inaccesible para todas las VPC. Utilice una política de recursos para conceder a las VPC y puntos de conexión de VPC acceso a las API privadas. Puede conceder acceso a un punto de conexión de VPC en cualquier cuenta de AWS.

La política de recursos debe contener condiciones `aws:SourceVpc` o `aws:SourceVpce` para restringir el acceso. Le recomendamos que identifique VPC y puntos de conexión de VPC específicos y no cree una política de recursos que permita el acceso a todas las VPC y puntos de conexión de VPC.

El siguiente procedimiento muestra cómo asociar una política de recursos a la API.

Consola de administración de AWS

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija una API de REST.
3. En el panel de navegación principal, elija Política de recursos.
4. Elija Crear política.
5. Elija Seleccionar una plantilla y, a continuación, elija VPC de origen.
6. Sustituya `{{vpcID}}` (incluidas las llaves) por el ID de la VPC.
7. Seleccione Save changes (Guardar cambios).

AWS CLI

El siguiente comando update-rest-api permite asociar una política de recursos a una API existente:

```
aws apigateway update-rest-api \
    --rest-api-id a1b2c3 \
    --patch-operations op=replace,path=/
policy,value='"{\"jsonEscapedPolicyDocument\"}"'
```

Es posible que también desee controlar qué recursos tienen acceso al punto de conexión de VPC. Para controlar qué recursos tienen acceso al punto de conexión de VPC, adjunte una política de punto de conexión al punto de conexión de VPC. Para obtener más información, consulte the section called “Uso de políticas de punto de conexión de VPC para API privadas”.

(Opcional) Asociación o desasociación de un punto de conexión de VPC con una API privada

Cuando asocia un punto de conexión de VPC con la API privada, API Gateway genera un nuevo registro de DNS de alias de Route 53. Puede usar este registro para invocar las API privadas del mismo modo que las API públicas sin anular un encabezado `Host` o pasar un encabezado `x-apigw-api-id`.

La URL base generada tiene el siguiente formato:

```
https://{rest-api-id}-{vpce-id}.execute-api.{region}.amazonaws.com/{stage}
```

Associate a VPC endpoint (Consola de administración de AWS)

Puede asociar un punto de conexión de VPC con la API privada cuando la cree o después de crearla. En el siguiente procedimiento, se muestra cómo asociar un punto de conexión de VPC con una API creada anteriormente.

Asociación de un punto de conexión de VPC con una API privada

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Elija la API privada.
3. En el panel de navegación principal, elija Política de recursos.
4. Edite su política de recursos para permitir las llamadas desde su punto de conexión de VPC adicional.

5. En el panel de navegación principal, elija Configuración de la API.
6. En la sección Detalles de la API, elija Editar.
7. En ID de punto de conexión de VPC, seleccione ID de punto de conexión de VPC adicionales.
8. Seleccione Save.
9. Vuelva a implementar la API para que los cambios se apliquen.

Dissociate a VPC endpoint (Consola de administración de AWS)

Para desasociar un punto de conexión de VPC de una API de REST privada

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Elija la API privada.
3. En el panel de navegación principal, elija Política de recursos.
4. Edite su política de recursos para eliminar las menciones del punto de conexión de VPC que quiere desasociar de su API privada.
5. En el panel de navegación principal, elija Configuración de la API.
6. En la sección Detalles de la API, elija Editar.
7. Para los ID de punto de conexión de VPC, elija la X para desasociar el punto de conexión de VPC.
8. Seleccione Save.
9. Vuelva a implementar la API para que los cambios se apliquen.

Associate a VPC endpoint (AWS CLI)

El siguiente comando create-rest-api permite asociar puntos de conexión de VPC en el momento de creación de la API:

```
aws apigateway create-rest-api \
    --name Petstore \
    --endpoint-configuration '{ "types": ["PRIVATE"], "vpcEndpointIds" :
 ["vpce-0212a4ababd5b8c3e", "vpce-0393a628149c867ee"] }' \
    --region us-west-2
```

El resultado será similar al siguiente:

```
{
    "apiKeySource": "HEADER",
    "endpointConfiguration": {
        "types": [
            "PRIVATE"
        ],
        "vpcEndpointIds": [
            "vpce-0212a4ababd5b8c3e",
            "vpce-0393a628149c867ee"
        ]
    },
    "id": "u67n3ov968",
    "createdDate": 1565718256,
    "name": "Petstore"
}
```

El siguiente comando update-rest-api permite asociar puntos de conexión de VPC a una API ya creada:

```
aws apigateway update-rest-api \
    --rest-api-id u67n3ov968 \
    --patch-operations "op='add',path='/endpointConfiguration/
vpcEndpointIds',value='vpce-01d622316a7df47f9'" \
    --region us-west-2
```

El resultado será similar al siguiente:

```
{
    "name": "Petstore",
    "apiKeySource": "1565718256",
    "tags": {},
    "createdDate": 1565718256,
    "endpointConfiguration": {
        "vpcEndpointIds": [
            "vpce-0212a4ababd5b8c3e",
            "vpce-0393a628149c867ee",
            "vpce-01d622316a7df47f9"
        ],
        "types": [
            "PRIVATE"
        ]
```

```
    },
    "id": "u67n3ov968"
}
```

Vuelva a implementar la API para que los cambios se apliquen.

Disassociate a VPC endpoint (AWS CLI)

El siguiente comando update-rest-api permite desasociar un punto de conexión de VPC de una API privada:

```
aws apigateway update-rest-api \
    --rest-api-id u67n3ov968 \
    --patch-operations "op='remove',path='/endpointConfiguration/
vpcEndpointIds',value='vpce-0393a628149c867ee'" \
    --region us-west-2
```

El resultado será similar al siguiente:

```
{
    "name": "Petstore",
    "apiKeySource": "1565718256",
    "tags": {},
    "createdDate": 1565718256,
    "endpointConfiguration": {
        "vpcEndpointIds": [
            "vpce-0212a4ababd5b8c3e",
            "vpce-01d622316a7df47f9"
        ],
        "types": [
            "PRIVATE"
        ]
    },
    "id": "u67n3ov968"
}
```

Vuelva a implementar la API para que los cambios se apliquen.

Paso 4: Implementación de una API privada

Para implementar la API, debe crear una implementación de API y asociarla con una etapa. El siguiente procedimiento muestra cómo implementar la API privada.

Consola de administración de AWS

Implementación de una API privada

1. Elija la API.
2. Elija Implementar API.
3. En Etapa, seleccione Nueva etapa.
4. En Nombre de etapa, ingrese un nombre de etapa.
5. (Opcional) En Description (Descripción), introduzca una descripción.
6. Elija Implementar.

AWS CLI

El siguiente comando create-deployment permite implementar una API privada:

```
aws apigateway create-deployment --rest-api-id a1b2c3 \
  --stage-name test \
  --stage-description 'Private API test stage' \
  --description 'First deployment'
```

Solución de problemas de la API privada

A continuación, se le proporcionan consejos para solucionar errores y problemas que es posible que se encuentre al crear una API privada.

Problema: No puedo conectarme a mi API pública desde un punto de conexión de VPC de API Gateway

Al crear la VPC, puede configurar los ajustes de DNS. Le recomendamos que active el DNS privado para la VPC. Si elige desactivar el DNS privado, solo puede acceder a la API a través de DNS público.

Si habilita DNS privado, no puede acceder al punto de conexión predeterminado de una API de API Gateway pública desde el punto de conexión de VPC. Puede acceder a una API con un nombre de dominio personalizado.

Si crea un nombre de dominio personalizado regional, utilice un registro de alias de tipo A. Si crea un nombre de dominio personalizado optimizado para sistemas perimetrales, no hay restricciones para

el tipo de registro. Puede acceder a estas API públicas con el DNS privado habilitado. Para obtener más información, consulte Problema: Me conecto a mi API pública desde un punto de conexión de VPC de API Gateway.

Problema: mi API devuelve **`{"Message":"User: anonymous is not authorized to perform: execute-api:Invoke on resource: arn:aws:execute-api:us-east-1:********/****/****/"}`**

En la política de recursos, si establece la entidad principal en una entidad principal de AWS, por ejemplo:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:role/developer",
                    "arn:aws:iam::111122223333:role/Admin"
                ]
            },
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/stage/GET/pets"
            ]
        }
    ]
}
```

Debe usar la autorización de `AWS_IAM` para todos los métodos de la API o, de lo contrario, la API devolverá el mensaje de error anterior. Para obtener más instrucciones sobre cómo activar la autorización de `AWS_IAM` para un método, consulte the section called “Métodos”.

Problema: No puedo saber si mi punto de conexión de VPC está asociado a mi API

Si asocia o desasocia un punto de conexión de VPC con la API privada, tendrá que volver a implementar la API. Es posible que la operación de actualización tarde unos minutos en completarse

debido a la propagación de DNS. Durante este tiempo la API esta disponible, pero es posible que mientras se esté realizando la propagación de DNS para las URL de DNS recién generadas. Si, después de varios minutos, las nuevas URL siguen sin resolverse en DNS, le recomendamos que vuelva a implementar la API.

Nombres de dominio personalizados para API privadas en API Gateway

Puede crear un nombre de dominio personalizado para sus API privadas. Use un nombre de dominio personalizado privado para proporcionar a los intermediarios de la API una URL más sencilla e intuitiva. Con un nombre de dominio personalizado privado, puede reducir la complejidad, configurar medidas de seguridad durante el protocolo de enlace TLS y controlar el ciclo de vida de los certificados de su nombre de dominio mediante AWS Gestor de certificados (ACM). Para obtener más información, consulte the section called “Protección de la clave privada de certificado de un nombre de dominio personalizado”.

Los nombres de dominio personalizados de API privadas no necesitan ser exclusivos entre distintas cuentas. Puede crear `example.private.com` en la cuenta 111122223333 y en la cuenta 555555555555, siempre y cuando el certificado ACM cubra el nombre de dominio. Para identificar un nombre de dominio personalizado privado, utilice el ARN del nombre de dominio personalizado privado. Este identificador es exclusivo de los nombres de dominio personalizados privados.

Cuando crea un nombre de dominio personalizado privado en API Gateway, usted es un proveedor de API. Puede proporcionar su nombre de dominio personalizado privado a otras Cuentas de AWS mediante API Gateway o AWS Resource Access Manager (AWS RAM).

Cuando invoca un nombre de dominio personalizado privado, usted es un consumidor de API. Puede consumir un nombre de dominio personalizado privado de su propia Cuenta de AWS o de otra Cuenta de AWS.

Cuando consume un nombre de dominio personalizado privado, crea una asociación de acceso a nombre de dominio entre un punto de conexión de VPC y un nombre de dominio personalizado privado. Con una asociación de acceso a nombre de dominio, los consumidores de la API pueden invocar su nombre de dominio personalizado privado mientras están aislados de la Internet pública. Para obtener más información, consulte the section called “Proveedores de API y consumidores de API”.

Protección de la clave privada de certificado de un nombre de dominio personalizado

Cuando solicita un certificado SSL/TLS con ACM para crear un nombre de dominio personalizado para API privadas, ACM genera un par de clave pública-privada. Cuando importa un certificado,

es usted quien genera el par de claves. La clave pública pasa a formar parte del certificado. Para almacenar la clave privada de forma segura, ACM crea otra clave utilizando AWS KMS, denominada clave de KMS, con el alias aws/acm. AWS KMS utiliza esta clave para cifrar la clave privada del certificado. Para obtener más información, consulte Protección de datos en AWS Gestor de certificados en la Guía del usuario de AWS Gestor de certificados.

API Gateway usa AWS TLS Connection Manager, un servicio al que solo pueden acceder los Servicios de AWS, para proteger y usar las claves privadas de su certificado. Cuando usa su certificado de ACM para crear un nombre de dominio personalizado de API Gateway, API Gateway asocia su certificado a AWS TLS Connection Manager. Para ello, creamos una concesión en AWS KMS para su clave gestionada de AWS. Esta concesión permite que TLS Connection Manager utilice AWS KMS para descifrar la clave privada de su certificado. TLS Connection Manager usa el certificado y la clave privada descifrada (texto sin formato) para establecer una conexión segura (sesión SSL/TLS) con los clientes de los servicios de API Gateway. Cuando el certificado se desvincula de un servicio de API Gateway, la concesión se retira. Para obtener más información, consulte Concesiones en la Guía para desarrolladores de AWS Key Management Service.

Para obtener más información, consulte the section called “Cifrado en reposo”.

Consideraciones sobre los nombres de dominio personalizados privados

Las siguientes consideraciones pueden afectar al uso de los nombres de dominio personalizados privados:

- API Gateway tarda unos 15 minutos en aprovisionar el nombre de dominio personalizado privado.
- Si actualiza el certificado de ACM, API Gateway tarda unos 15 minutos en completar la actualización. Durante este tiempo, el nombre de dominio estará tiene el estado `UPDATING`, y podrá seguir accediendo a él.
- Para invocar un nombre de dominio personalizado privado, debe crear una asociación de acceso a nombre de dominio. Tras crear una asociación de acceso a nombre de dominio, esta tardará unos 15 minutos en estar lista.
- El ARN de nombre de dominio personalizado privado contiene *`account-id`* y *`domain-name-id`*. Cuando crea un nombre de dominio, API Gateway utiliza el formato de ARN de `arn:`*`partition`*`:apigateway:`*`region`*`::/domainnames/domain-name`. Cuando accede a un nombre de dominio personalizado privado, utiliza el formato de ARN de `arn:`*`partition`*`:apigateway:`*`region`*`:`*`account-id`*`:/domainnames/domain-name+`*`domain-name-id`*.

Es posible que tenga que modificar los permisos de IAM para permitir el acceso a un nombre de dominio privado después de crearlo.

- No puede invocar nombres de dominio personalizados privados con el mismo nombre desde el mismo punto de conexión de VPC. Por ejemplo, si desea invocar `arn:aws:apigateway:us-west-2:111122223333:/domainnames/private.example.com+abcd1234` y `arn:aws:apigateway:us-west-2:111122223333:/domainnames/private.example.com+xyz000`, asocie cada nombre de dominio personalizado privado a un punto de conexión de VPC distinto.
- Se admiten los certificados comodín (por ejemplo, un certificado para `*.private.example.com`).
- Los nombres de dominio personalizados comodín no se admiten.
- Solo se admiten los certificados RSA con una longitud de clave de 2048 bits y los certificados ECDSA con longitudes de clave de 256 y 384 bits.
- No puede configurar el tipo de dirección IP de las API privadas para permitir que solo las direcciones IPv4 invoquen la API privada. Solo se admite pila doble. Para obtener más información, consulte the section called “Tipos de direcciones IP para API de REST en API Gateway”.
- Para enviar tráfico con la API privada, puede usar todos los tipos de direcciones IP compatibles con Amazon VPC. Puede enviar tráfico de pila doble e IPv6 configurando los ajustes en el punto de conexión de VPC. No puede modificarlo con API Gateway. Para obtener más información, consulte Adición de la compatibilidad de IPv6 con su VPC.
- No se admite la asignación de rutas base multinivel, como asignar la API privada a `/developers/feature`, pero puede utilizar una regla de enrutamiento para crear una condición de ruta multinivel. Para obtener más información, consulte the section called “Envío de tráfico a las API a través del nombre de dominio personalizado en API Gateway”.
- No puede establecer una versión de TLS mínima para su nombre de dominio personalizado privado. Todos los nombres de dominio personalizados privados tienen la política de seguridad de `TLS-1-2`.
- Puede utilizar la política de punto de conexión de VPC para controlar el acceso a un nombre de dominio personalizado privado. Para obtener más información, consulte los ejemplos 4 y 5 en the section called “Uso de políticas de punto de conexión de VPC para API privadas”.
- Debe crear una política de recursos independiente para la API privada y para el nombre de dominio personalizado privado. Para invocar un nombre de dominio personalizado privado, un consumidor de API necesita acceder desde la política de recursos de nombre de dominio

personalizado privado, la política de recursos de API privada y cualquier política de punto de conexión de VPC o autorización en la API privada.

Consideraciones sobre el uso de nombres de dominio personalizados privados con otros recursos de API Gateway

Las siguientes consideraciones pueden afectar a la forma en que se utilizan los nombres de dominio personalizados privados con otros recursos de API Gateway:

- No puede enviar tráfico desde un nombre de dominio personalizado privado a una API pública.
- Cuando una API privada se asigna a un nombre de dominio personalizado privado, no es posible cambiar el tipo de punto conexión de la API.
- No es posible migrar un nombre de dominio personalizado público a un nombre de dominio personalizado privado.
- Si tiene un punto de conexión de VPC que utiliza para acceder a un nombre de dominio personalizado público, no lo utilice para crear una asociación de acceso a nombres de dominio con un nombre de dominio personalizado privado.

Diferencias entre los nombres de dominio personalizados privados y los nombres de dominio personalizados públicos

A continuación se describen las diferencias que existen entre los nombres de dominio personalizados privados y públicos:

- Los nombres de dominio personalizados privados no tienen por qué ser exclusivos entre distintas cuentas.
- Un nombre de dominio privado tiene un ID de nombre de dominio. Este ID solo identifica un nombre de dominio personalizado privado y no se genera para los nombres de dominio personalizados públicos.
- Cuando utilice la AWS CLI para actualizar o eliminar su nombre de dominio personalizado privado, deberá proporcionar el ID del nombre de dominio. Si tiene un nombre de dominio personalizado privado que se llama `example.com` y un nombre de dominio personalizado público que se llama `example.com` y no proporciona el ID del nombre de dominio, API Gateway modificará o eliminará el nombre de dominio personalizado público.

Pasos siguientes para los nombres de dominio personalizados de API privadas

Para obtener información sobre las tareas de un proveedor de API y un consumidor de API, consulte the section called “Proveedores de API y consumidores de API”.

Para obtener instrucciones sobre cómo crear un nombre de dominio personalizado privado que pueda invocar en su propia Cuenta de AWS, consulte the section called “Tutorial: Creación e invocación de un nombre de dominio personalizado para API privadas”.

Para obtener instrucciones sobre cómo proporcionar acceso a otra Cuenta de AWS a su nombre de dominio personalizado privado, consulte the section called “Proveedor de API: compartir su nombre de dominio personalizado privado mediante AWS RAM”. Para obtener instrucciones sobre cómo asociar su punto de conexión de VPC a un nombre de dominio personalizado privado de otra Cuenta de AWS, consulte the section called “Consumidor de API: asociar su punto de conexión de VPC a un nombre de dominio personalizado privado que se comparta con usted”.

Tareas de los proveedores de API y de los consumidores de API para los nombres de dominio personalizados de API privadas

Cuando crea un nombre de dominio personalizado privado, usted es un proveedor de API. Cuando invoca un nombre de dominio personalizado privado, usted es un consumidor de API. Puede consumir un nombre de dominio personalizado privado de su propia Cuenta de AWS o de otra Cuenta de AWS.

En la siguiente sección se explican las tareas que deben realizar el proveedor de API y el consumidor de API para utilizar un nombre de dominio personalizado privado. Si desea invocar un nombre de dominio personalizado privado de su propia Cuenta de AWS, será tanto el proveedor como el consumidor de la API. Si desea invocar un dominio personalizado privado de otra Cuenta de AWS, en función de la relación de confianza entre el proveedor de API y el consumidor de API en AWS Organizations, es posible que AWS RAM deba llevar a cabo una serie de tareas en su nombre.

Tareas de un proveedor de API

Los proveedores de API crean API privadas y las asignan a nombres de dominio personalizados.

Los proveedores de API administran dos políticas de recursos para proteger sus nombres de dominio personalizados privados. La primera política es para el servicio `execute-api` y controla qué puntos de conexión de VPC pueden invocar su nombre de dominio personalizado privado. En la configuración del nombre de dominio personalizado privado, se denomina "`policy`".

La segunda política es para el servicio Amazon API Gateway Management y controla qué puntos de conexión de VPC de otras Cuentas de AWS pueden formar una asociación de acceso a nombres de dominio con su nombre de dominio personalizado privado. Un punto de conexión de VPC debe formar una asociación de acceso a nombres de dominio con un nombre de dominio privado personalizado para invocarlo. En la configuración del nombre de dominio personalizado privado, es la `managementPolicy`. Puede usar AWS RAM o API Gateway para actualizar esta política. Si no tiene previsto permitir que los puntos de conexión de VPC de otras Cuentas de AWS invoquen su nombre de dominio personalizado, no edite la `managementPolicy`.

Si es un proveedor de API, debe hacer lo siguiente:

1. Cree una API privada.
2. Actualice su `policy` de API privadas para conceder a su punto de conexión de VPC acceso a su API privada.
3. Cree un nombre de dominio personalizado privado.
4. Actualice su `policy` de nombre de dominio personalizado privado para conceder a su punto de conexión de VPC acceso a su nombre de dominio personalizado privado.
5. Cree una asignación de ruta base o una regla de enrutamiento para enviar tráfico desde la API privada al nombre de dominio personalizado privado. Para obtener más información, consulte the section called “Envío de tráfico a las API a través del nombre de dominio personalizado en API Gateway”.

Si desea permitir que los consumidores de API de otras Cuentas de AWS accedan a su nombre de dominio personalizado privado, haga lo siguiente:

1. Actualice la `managementPolicy` de su nombre de dominio personalizado privado para permitir que los consumidores de API de otras cuentas asocien sus puntos de conexión de VPC a su nombre de dominio personalizado privado. Para ello, tiene los siguientes métodos:

   AWS RAM

   > Con AWS RAM, si el proveedor de API y el consumidor de API están en la misma organización que utiliza AWS Organizations, se acepta automáticamente el uso compartido de recursos entre el proveedor y el consumidor. De lo contrario, debe esperar a que el consumidor de API acepte el uso recurso compartido. Le recomendamos que utilice AWS RAM para compartir su nombre de dominio personalizado privado.

API Gateway

Con API Gateway, solo se admite la AWS CLI. Debe actualizar su nombre de dominio personalizado privado mediante una operación de parche y proporcionar su propio documento de política para la `managementPolicy`.

2. Actualice la `policy` de su nombre de dominio personalizado privado y cualquier API privada asignada a él para conceder acceso al punto de conexión de VPC del consumidor de API.

Para obtener instrucciones sobre cómo proporcionar su API a otra Cuenta de AWS, consulte the section called “Proveedor de API: compartir su nombre de dominio personalizado privado mediante AWS RAM”.

Tareas de un consumidor de API

Los consumidores de API asocian sus puntos de conexión de VPC a un ARN de nombre de dominio para invocar un nombre de dominio personalizado privado. Los consumidores de API no necesitan crear una API de API Gateway.

Si es un consumidor de API, haga lo siguiente:

1. Cree un punto de conexión de VPC con un DNS privado habilitado en Amazon VPC.
2. (Opcional, si se utiliza AWS RAM) Acepte un recurso compartido de dominio personalizado privado en AWS RAM en un plazo de 12 horas a partir del momento en que se haya compartido el recurso. Si usted y el proveedor de API forman parte de la misma organización, el recurso compartido se acepta automáticamente.
3. Obtenga el ARN del nombre de dominio personalizado privado. Como la URL del nombre de dominio personalizado privado no es exclusiva, se utiliza el ARN del nombre de dominio personalizado privado para formar la asociación de acceso a nombres de dominio entre el punto de conexión de VPC y el nombre de dominio personalizado privado. Puede utilizar AWS RAM para recuperar el ARN del nombre de dominio personalizado privado.
4. Asocie el ARN del dominio personalizado privado a su punto de conexión de VPC en API Gateway. Esto crea una conexión segura entre el punto de conexión de VPC y el nombre de dominio personalizado privado. El tráfico no sale de la red de Amazon.
5. Espere a que el proveedor de API conceda a su punto de conexión de VPC acceso al nombre de dominio personalizado privado y a cualquier API privada asignada al nombre de dominio personalizado privado. Si es tanto proveedor como consumidor de la API, concederá acceso de invocación a su propio punto de conexión de VPC.

6. Cree una zona alojada privada de Route 53 y un registro de Route 53 para resolver el nombre de dominio personalizado privado en Route 53.

Para obtener instrucciones sobre cómo consumir una API en otra Cuenta de AWS, consulte the section called “Consumidor de API: asociar su punto de conexión de VPC a un nombre de dominio personalizado privado que se comparta con usted”.

Tutorial: Creación e invocación de un nombre de dominio personalizado para API privadas

En este tutorial, creará un nombre de dominio personalizado privado que podrá invocar en una VPC en su propia cuenta. Para ello, usted es tanto proveedor de API como consumidor de API. Necesita una API privada y un punto de conexión de VPC existentes para completar este tutorial. Si tiene un punto de conexión de VPC que utiliza para acceder a un nombre de dominio personalizado público, no lo utilice para este tutorial ni para crear asociaciones de acceso a nombres de dominio.

Paso 1: creación de un nombre de dominio personalizado privado

El nombre de dominio personalizado privado se crea especificando el nombre de dominio, el certificado de ACM y la política del servicio `execute-api` para controlar qué puntos de conexión de VPC pueden invocarlo.

Consola de administración de AWS

Para crear un nombre de dominio personalizado privado

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Nombres de dominio personalizados.
3. Elija Agregar nombre de dominio.
4. En Nombre de dominio, escriba un nombre de dominio.

   Su certificado de ACM debe cubrir este nombre de dominio, pero no es necesario que el nombre de dominio sea exclusivo.
5. Seleccione Privado.
6. En Modo de enrutamiento, elija Solo asignaciones de la API.
7. En Certificado de ACM, seleccione un certificado.
8. Elija Agregar nombre de dominio.

API Gateway proporciona un nombre de dominio con una política de `deny` de todos los recursos. Esta es la política de recursos del servicio `execute-api`. Debe actualizar esta política de recursos para conceder acceso a sus puntos de conexión de VPC para invocar su nombre de dominio personalizado privado.

Para actualizar la política de recursos

1. Elija la pestaña Política de recursos y, a continuación, seleccione Editar política de recursos.
2. Especifique la siguiente política de recursos en el editor de código. Sustituya el punto de conexión de VPC *vpce-abcd1234efg* por el ID de su propio punto de conexión de VPC.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ],
            "Condition" : {
                "StringNotEquals": {
                    "aws:SourceVpce": "vpce-abcd1234"
                }
            }
        }
    ]
}
```

3. Seleccione Save changes (Guardar cambios).

AWS CLI

Cuando crea un nombre de dominio personalizado privado con la AWS CLI, proporciona una política de recursos para que el servicio `execute-api` conceda acceso a los puntos de conexión de VPC para invocar su nombre de dominio personalizado privado mediante el parámetro `--policy file://policy.json`. Podrá modificar esta política más tarde.

En este ejemplo, asociará la siguiente política de recursos como `policy` al cargar parámetros desde un archivo. Copie y guarde este archivo como `policy.json`. Esta política solo permite el tráfico entrante a un nombre de dominio personalizado privado desde el punto de conexión de VPC *vpce-abcd1234efg*:

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ]
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ],
            "Condition" : {
                "StringNotEquals": {
                    "aws:SourceVpce": "vpce-abcd1234"
                }
            }
        }
    ]
}
```

El siguiente comando create-domain-name permite crear un nombre de dominio personalizado privado:

```
aws apigateway create-domain-name \
    --domain-name 'private.example.com' \
    --certificate-arn 'arn:aws:acm:us-west-2:111122223333:certificate/
a1b2c3d4-5678-90ab-cdef' \
    --security-policy 'TLS_1_2' \
    --endpoint-configuration '{"types":["PRIVATE"]}' \
    --policy file://policy.json
```

El resultado será similar al siguiente:

```
{
    "domainName": "private.example.com",
    "domainNameId": "abcd1234",
    "domainNameArn": "arn:aws:apigateway:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234",
    "certificateArn": "arn:aws:acm:us-west-2:111122223333:certificate/
a1b2c3d4-5678-90ab-cdef",
    "certificateUploadDate": "2024-09-10T10:31:20-07:00",
    "endpointConfiguration": {
        "types": [
            "PRIVATE"
        ]
    },
    "domainNameStatus": "AVAILABLE",
    "securityPolicy": "TLS_1_2",
    "routingMode" : "API_MAPPING_ONLY",
    "policy": "..."
}
```

Paso 2: creación de una asignación de ruta base para asignar la API privada al nombre de dominio personalizado privado

Después de crear el nombre de dominio personalizado privado, debe asignarle una API privada. Una asignación de ruta base hace que se pueda acceder a la API a través de una combinación de nombre de dominio personalizado privado y una ruta base asociada. Le recomendamos que utilice un único nombre de dominio personalizado privado como nombre de host de varias API privadas.

Todos los proveedores de API deben crear una asignación de ruta base, incluso si no tienen previsto invocar su propia API. También debe conceder acceso a los puntos de conexión de VPC para invocar cualquier API privada que asigne a su nombre de dominio personalizado privado.

Consola de administración de AWS

Para crear una asignación de ruta base

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Nombres de dominio personalizados.
3. Elija un nombre de dominio personalizado privado.
4. En la pestaña Asignaciones de API, elija Configurar asignaciones.
5. Seleccione Agregar nueva asignación.
6. Introduzca una API, una Etapa y, de forma opcional, una Ruta.
7. Seleccione Save.

AWS CLI

El siguiente comando create-base-path-mapping permite crear una asignación entre una API privada y un nombre de dominio personalizado privado:

```
aws apigateway create-base-path-mapping \
    --domain-name-id abcd1234 \
    --domain-name 'private.example.com' \
    --rest-api-id a1b2c3 \
    --stage prod \
    --base-path v1
```

El resultado será similar al siguiente.

```
{
    "basePath": "v1",
    "restApiId": "a1b2c3",
    "stage": "prod"
}
```

Para obtener mayor flexibilidad sobre cómo enrutar el tráfico a las API, puede cambiar el modo de enrutamiento a `ROUTING_RULE_ONLY` o `ROUTING_RULE_THEN_API_MAPPING` y crear una regla de

enrutamiento. Para obtener más información, consulte the section called “Envío de tráfico a las API a través del nombre de dominio personalizado en API Gateway”.

> **Note**
>
> Si desea que otras Cuentas de AWS invoquen su nombre de dominio personalizado privado, una vez que haya completado este tutorial, siga los pasos que se describen en the section called “Proveedor de API: compartir su nombre de dominio personalizado privado mediante AWS RAM”.

Paso 3: creación de una asociación de acceso a nombres de dominio entre su nombre de dominio personalizado y un punto de conexión de VPC

A continuación, debe crear una asociación de acceso a nombres de dominio entre su nombre de dominio personalizado privado y su punto de conexión de VPC. El punto de conexión de VPC utiliza la asociación de acceso a nombres de dominio para invocar su nombre de dominio personalizado privado mientras está aislado de la Internet pública.

Consola de administración de AWS

Para crear una asociación de acceso a nombres de dominio

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Nombres de dominio personalizados.
3. Elija un nombre de dominio personalizado privado.
4. En la pestaña Uso compartido de recursos, en Asociaciones de acceso a nombres de dominio, seleccione Crear asociación de acceso a nombres de dominio.
5. En ARN del nombre de dominio, seleccione el nombre de dominio.
6. En ID del punto de conexión de VPC, seleccione el ID del punto de conexión de VPC al que proporcionó acceso en el paso 1.
7. Elija Asociación de acceso a nombres de dominio.

También puede crear su asociación de acceso a nombres de dominio desde la página Asociaciones de acceso a nombres de dominio de la consola.

AWS CLI

El siguiente comando `create-domain-name-access-association` permite crear una asociación de acceso a nombres de dominio entre su nombre de dominio personalizado privado y su punto de conexión de VPC.

```
aws apigateway create-domain-name-access-association \
    --domain-name-arn arn:aws:apigateway:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234 \
    --access-association-source vpce-abcd1234efg \
    --access-association-source-type VPCE \
    --region us-west-2
```

El resultado será similar al siguiente.

```
{
    "domainNameAccessAssociationARN": "arn:aws:apigateway:us-west-2:111122223333:/
domainnameaccessassociations/domainname/private.example.com+abcd1234/vpcesource/
vpce-abcd1234efg",
    "accessAssociationSource": "vpce-abcd1234efg",
    "accessAssociationSourceType": "VPCE",
    "domainNameARN" : "arn:aws:apigateway:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234"
}
```

Tras crear la asociación de acceso a nombres de dominio, esta tardará unos 15 minutos en estar lista. Mientras espera, puede continuar con los siguientes pasos.

Paso 4: creación de una zona alojada en Route 53

Tras actualizar la política de recursos y asociar el nombre de dominio personalizado privado al punto de conexión de VPC, debe crear una zona alojada privada en Route 53 para resolver el nombre de dominio personalizado. Una zona alojada es un contenedor que incluye información acerca de cómo se desea dirigir el tráfico de un dominio dentro de una o varias VPC sin tener que exponer sus recursos en Internet. Para obtener más información, consulte Uso de zonas alojadas privadas.

Consola de administración de AWS

Para usar la Consola de administración de AWS, consulte Creación de una zona alojada privada en la Guía para desarrolladores de Amazon Route 53.

En Nombre, escriba el nombre de su nombre de dominio personalizado privado. En ID de VPC, utilice la VPC que contiene el punto de conexión de VPC que utilizó en los pasos anteriores.

AWS CLI

El siguiente comando create-hosted-zone permite crear una zona alojada privada:

```
aws route53 create-hosted-zone --name private.example.com \
    --caller-reference 2014-04-01-18:47 \
    --hosted-zone-config Comment="command-line version",PrivateZone=true \
    --vpc VPCRegion=us-west-2,VPCId=vpc-abcd1234
```

El resultado contiene el ID de la zona alojada. El ID de la zona alojada se utiliza en los pasos siguientes.

Paso 5: creación de un registro de DNS de Route 53

Después de crear la zona alojada, debe crear un registro para resolver el nombre de dominio personalizado privado. Debe utilizar el ID de zona alojada que creó en el paso anterior. En este ejemplo, se crea un tipo de registro A. Si está utilizando IPv6 para el punto de conexión de VPC, cree un tipo de registro AAAA. Si utiliza la pila doble para el punto de conexión de VPC, cree un tipo de registro AAAA y un tipo de registro A.

Consola de administración de AWS

Para usar la Consola de administración de AWS, consulte Enrutamiento del tráfico a una API de Amazon API Gateway mediante su nombre de dominio.

Utilice Creación rápida y active la opción Alias. Para el punto de conexión, utilice el nombre de DNS del punto de conexión de VPC.

AWS CLI

Para configurar sus registros de DNS para asignar el nombre de dominio personalizado privado al nombre de host del ID de zona alojada designado, cree un archivo JSON que contenga la configuración a fin de establecer un registro de DNS para el nombre de dominio privado.

El siguiente `setup-dns-record.json` muestra cómo crear un registro `A` de DNS para asignar un nombre de dominio personalizado privado a su nombre de host privado. Debe proporcionar el `DNSName` de su ID de DNS de VPC y el ID de la zona alojada que creó en el paso anterior.

```
{
```

```
  "Changes": [
    {
      "Action": "UPSERT",
      "ResourceRecordSet": {
        "Name": "private.example.com",
        "Type": "A",
        "AliasTarget": {
          "DNSName": "vpce-abcd1234.execute-api.us-west-2.vpce.amazonaws.com",
          "HostedZoneId": "Z2OJLYMUO9EFXC",
          "EvaluateTargetHealth": false
        }
      }
    }
  ]
}
```

El siguiente comando change-resource-record-sets permite crear un registro de DNS para su nombre de dominio personalizado privado:

```
aws route53 change-resource-record-sets \
    --hosted-zone-id ZABCDEFG1234 \
    --change-batch file://path/to/your/setup-dns-record.json
```

Reemplace `hosted-zone-id` por el ID de zona alojada de Route 53 del conjunto de registros de DNS de la cuenta. El valor del parámetro `change-batch` apunta al archivo JSON.

Si no tiene previsto invocar su propio nombre de dominio personalizado privado, una vez que haya confirmado que el nombre de dominio personalizado privado funciona, puede eliminar estos recursos.

Paso 6: invocación del nombre de dominio personalizado privado

Ahora puede invocar su nombre de dominio personalizado privado en su propia Cuenta de AWS. En su VPC, utilice el siguiente comando curl para acceder a su nombre de dominio personalizado privado.

```
curl https://private.example.com/v1
```

Para obtener más información sobre otras formas de invocar sus API privadas, consulte the section called “Invocación de una API privada con un nombre de dominio personalizado”.

Paso 7: limpiar

Para evitar costos innecesarios, elimine la asociación entre el punto de conexión de VPC y el nombre de dominio personalizado privado y, a continuación, elimine el nombre de dominio personalizado privado.

Consola de administración de AWS

Para eliminar la asociación de acceso a nombres de dominio

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Asociaciones de acceso a nombres de dominio.
3. Seleccione la asociación de acceso a nombres de dominio y, a continuación, elija Eliminar.
4. Confirme la selección y, a continuación, elija Eliminar.

Después de eliminar la asociación de acceso a nombres de dominio, puede eliminar el nombre de dominio personalizado privado.

Para eliminar un nombre de dominio personalizado privado

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Nombres de dominio personalizados.
3. Elija el nombre de dominio personalizado privado.
4. Elija Eliminar.
5. Confirme la selección y, a continuación, elija Eliminar.

Si es necesario, también puede eliminar el punto de conexión de VPC. Para obtener más información, consulte Eliminación de un punto de conexión de interfaz.

AWS CLI

Eliminación

1. El siguiente comando `delete-access-association` permite eliminar la asociación de acceso a nombres de dominio:

```
aws apigateway delete-domain-name-access-association \
```

```
    --domain-name-access-association-arn 'arn:aws:apigateway:us-
west-2:111122223333:/domainnameaccessassociations/domainname/private.example.com
+abcd1234/vpcesource/vpce-abcd1234efg' \
    --region us-west-2
```

2. El siguiente comando `delete-domain-name` permite eliminar el nombre de dominio personalizado privado. Este comando también elimina todas las asignaciones de ruta base.

```
aws apigateway delete-domain-name \
    --domain-name test.private.com \
    --domain-name-id abcd1234
```

Si es necesario, también puede eliminar el punto de conexión de VPC. Para obtener más información, consulte Eliminación de un punto de conexión de interfaz.

Prácticas recomendadas

Le recomendamos que utilice las siguientes prácticas recomendadas al crear el nombre de dominio personalizado privado:

- Utilice la asignación de rutas base o las reglas de enrutamiento para enviar tráfico desde un nombre de dominio personalizado privado a varias API privadas.
- Cuando un punto de conexión de VPC ya no necesite acceder a un nombre de dominio personalizado privado, elimine la asociación. Además, elimine el punto de conexión de VPC de la `policy` del servicio `execute-api` para el dominio personalizado privado.
- Configure al menos dos zonas de disponibilidad por punto de conexión de VPC.
- Deshabilite el punto de conexión predeterminado. Le recomendamos que deshabilite el punto de conexión predeterminado para permitir que los consumidores de API solo puedan llamar a su API desde el nombre de dominio personalizado. Para obtener más información, consulte the section called “Deshabilitación del punto de conexión predeterminado”.
- Le recomendamos que aprovisione una zona alojada privada de Route 53 y un registro de tipo A cuando configure su nombre de dominio personalizado privado. Si no tiene previsto invocar su propio nombre de dominio personalizado privado, puede eliminar estos recursos más tarde.

Uso de nombres de dominio personalizados privados entre cuentas

En esta sección se explica cómo usar nombres de dominio personalizados privados entre cuentas. Puede proporcionar un nombre de dominio personalizado privado a otra Cuenta de AWS y usar otra Cuenta de AWS para invocar un nombre de dominio personalizado privado.

Puede compartir su nombre de dominio personalizado privado con otra Cuenta de AWS mediante AWS Resource Access Manager o API Gateway. AWS Resource Access Manager (AWS RAM) le ayuda a compartir sus recursos de forma segura entre Cuentas de AWS y dentro de su organización o de unidades organizativas (OU). Para obtener más información, consulte ¿Qué es AWS Resource Access Manager?

Para obtener instrucciones sobre cómo compartir un nombre de dominio personalizado privado con otra Cuenta de AWS mediante AWS RAM, consulte the section called “Proveedor de API: compartir su nombre de dominio personalizado privado mediante AWS RAM”.

Para obtener instrucciones sobre cómo compartir un nombre de dominio personalizado privado con otra Cuenta de AWS mediante API Gateway, consulte the section called “Proveedor de API: compartir su nombre de dominio personalizado privado mediante la de API Gateway AWS CLI”.

Para obtener instrucciones sobre cómo consumir un nombre de dominio personalizado privado en otra Cuenta de AWS, consulte the section called “Consumidor de API: asociar su punto de conexión de VPC a un nombre de dominio personalizado privado que se comparta con usted”.

Prácticas recomendadas para el uso de nombres de dominio personalizados privados entre cuentas

Recomendamos utilizar las siguientes prácticas recomendadas para usar con nombres de dominio personalizados privados entre cuentas:

- Utilice AWS RAM para compartir sus nombres de dominio personalizados privados. Al usar AWS RAM, podrá reducir la sobrecarga operativa y no tendrá que crear una `managementPolicy` para el servicio Amazon API Gateway Management.
- Utilice el parámetro `resource-owner` cuando enumere sus nombres de dominio personalizados privados o asociaciones de acceso a nombres de dominio. Utilice el parámetro `resource-owner` para enumerar únicamente los recursos que le pertenecen a usted o a otras Cuentas de AWS.

  En el siguiente ejemplo se muestra cómo obtener todas las asociaciones de acceso a nombres de dominio que posee:

```
aws apigateway get-domain-name-access-associations --resource-owner SELF
```

Utilice `--resource-owner OTHER_ACCOUNTS` para enumerar todas las asociaciones de acceso a nombres de dominio que otras cuentas han formado con su nombre de dominio personalizado privado.

Proveedor de API: compartir su nombre de dominio personalizado privado mediante AWS RAM

Puede proporcionar acceso a su nombre de dominio personalizado privado a los consumidores de API de otras Cuentas de AWS. En esta sección, aprenderá a compartir su nombre de dominio personalizado privado mediante AWS RAM y a controlar el acceso al nombre de dominio personalizado privado.

Consideraciones a la hora de compartir su nombre de dominio personalizado privado

Las siguientes consideraciones pueden afectar a la forma en que proporciona acceso a su nombre de dominio personalizado privado mediante AWS RAM. Para obtener información sobre cómo compartir su nombre de dominio personalizado privado sin usar AWS RAM, consulte the section called “Proveedor de API: compartir su nombre de dominio personalizado privado mediante la de API Gateway AWS CLI”.

- Los nombres de dominio personalizados privados se comparten en el nivel de Región de AWS. Tanto el nombre de dominio personalizado privado como el punto de conexión de VPC deben pertenecer a la misma Región de AWS.
- Puede usar un recurso compartido con varias entidades principales y, una vez creado el recurso compartido, puede agregarle más entidades principales. Le recomendamos que reutilice el recurso compartido siempre que sea posible.
- Debe conceder siempre acceso al punto de conexión de VPC del consumidor de API para invocar el nombre de dominio personalizado privado y cualquier API privada asignada a él.
- Si el consumidor de API y el proveedor de API pertenecen a la misma organización que utiliza AWS Organizations, el recurso compartido se acepta automáticamente. Aún en este caso debe crear el recurso compartido mediante AWS RAM.
- Si el consumidor de API y el proveedor de API pertenecen a la misma organización que utiliza AWS Organizations y la organización tiene habilitado el uso compartido de recursos, se concederá automáticamente acceso a los recursos compartidos a todas las entidades principales de la organización con las que comparte. No se precisa invitación y puede omitir el uso compartido de recursos.

- Si el consumidor de API no acepta el recurso compartido en un plazo de 12 horas, el proveedor de API debe volver a compartir el recurso.
- Tras crear el recurso compartido, AWS RAM actualiza la `managementPolicy` del servicio Amazon API Gateway Management para su nombre de dominio personalizado privado a fin de impedir el acceso a las entidades principales que no dispongan de un acceso `allow` explícito. Para obtener más información, consulte Cómo determinar si una solicitud se permite o se deniega dentro de una cuenta en la Guía del usuario de IAM.

  La `managementPolicy` actualizada será similar a la siguiente:

  JSON

```
{
    "Version":"2012-10-17",
    "Id": "abcd1234-1234-abcd-abcd-1234abcdefg",
    "Statement": [
        {
            "Sid":
 "APIGatewayPrivateDomainNameManagementPolicyDefaultPermission-org",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "apigateway:CreateAccessAssociation",
            "Resource": "arn:aws:apigateway:us-west-2:111122223333:/
domainnames/private.example.com+abcd1234",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalOrgID": "o-1234abcd"
                },
                "StringNotEquals": {
                    "aws:PrincipalAccount": "111122223333"
                }
            }
        }
    ]
}
```

  AWS RAM ha impedido que las entidades principales sin acceso `allow` explícito creen asociaciones de acceso con su nombre de dominio personalizado privado al agregar lo siguiente:

```
"StringNotEquals": {
    "aws:PrincipalAccount": "111122223333"
```

```
}
```

Aun así, puede utilizar la entidad principal de la Cuenta de AWS que creó el nombre de dominio personalizado privado para crear asociaciones de acceso a nombres de dominio.

Permiso para que otras cuentas creen asociaciones de acceso a nombres de dominio con su nombre de dominio personalizado privado

En primer lugar, debe conceder acceso a otra Cuenta de AWS para crear asociaciones de acceso a nombres de dominio con su nombre de dominio personalizado privado.

Consola de administración de AWS

Para usar la Consola de administración de AWS, consulte Creación de un recurso compartido en AWS RAM en la Guía del usuario de AWS RAM.

En Seleccionar tipo de recurso, elija Dominios personalizados privados de API Gateway.

AWS CLI

El siguiente comando create-resource-share permite crear un recurso compartido para el nombre de dominio personalizado privado. La asociación del recurso y la entidad principal puede tardar unos minutos en completarse. Para las entidades principales, proporcione un ID de cuenta o un ID de Organizations, como `arn:aws:organizations::123456789012:organization/o-1234abcd`. Puede proporcionar varias entidades principales para el recurso compartido.

```
aws ram create-resource-share \
    --region us-west-2 \
    --name privateCustomDomain-resource-share \
    --permission-arns arn:aws:ram::aws:permission/
APIGatewayPrivateDomainNameManagementPolicyDefaultPermission \
    --resource-arns arn:aws:apigateway:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234 \
    --principals 222222222222
```

Una vez que haya proporcionado acceso a otra Cuenta de AWS, los consumidores de API de dicha cuenta deben crear una asociación de acceso a nombres de dominio entre su punto de conexión de VPC y su nombre de dominio personalizado privado. No puede crear la asociación de acceso a nombres de dominio para ellos. Para obtener más información, consulte the section

called “Asociación de su punto de conexión de VPC a un nombre de dominio personalizado privado compartido”.

Permiso para que otras cuentas invoquen su nombre de dominio personalizado privado

A continuación, debe conceder acceso al punto de conexión de VPC del consumidor de API para que invoque su nombre de dominio personalizado privado y cualquier API privada asignada a él.

Consola de administración de AWS

Para permitir que los puntos de conexión de VPC de otras cuentas invoquen su nombre de dominio personalizado privado

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Nombres de dominio personalizados.
3. Elija el nombre de dominio personalizado privado que compartió con otras Cuentas de AWS.
4. En la pestaña Política de recursos, elija Editar política de recursos.
5. Agregue el ID del punto de conexión de VPC del consumidor de API a la política de recursos.

   Puede encontrar el ID del punto de conexión de VPC del consumidor de API en la sección Asociaciones de acceso a nombres de dominio de la pestaña Uso compartido de recursos de la página Detalles del dominio de su nombre de dominio personalizado privado.
6. Seleccione Save changes (Guardar cambios).

AWS CLI

La siguiente `policy` del servicio `execute-api` permite el tráfico entrante a un nombre de dominio personalizado privado desde el punto de conexión de VPC *`vpce-abcd1234efg`* y *`vpce-xyz000abc`*.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
```

```
            ]
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "execute-api:/*"
            ],
            "Condition" : {
                "StringNotEquals": {
                    "aws:SourceVpce": [
                    "vpce-abcd1234",
                    "vpce-xyzz0000"
                    ]
                }
            }
        }
    ]
}
```

El siguiente comando update-domain-name utiliza una operación de parche para actualizar la `policy` para un nombre de dominio personalizado privado:

```
aws apigateway update-domain-name
    --domain-name private.example.com \
    --domain-name-id abcd1234 \
    --patch-operations op=replace,path=/policy,value='"{\"Version\": \"2012-10-17\",
       \"Statement\": [{\"Effect\": \"Allow\",\"Principal\": \"*\",\"Action\":
 \"execute-api:Invoke\",\"Resource\":[\"execute-api:/*\"]},{\"Effect\": \"Deny\",
\"Principal\": \"*\",\"Action\": \"execute-api:Invoke\",\"Resource\":[\"execute-
api:/*\"],\"Condition\":{\"StringNotEquals\":[\"vpce-abcd1234efg\", \"vpce-xyz000abc
\"]}}}]}"
```

Proveedor de API: cómo dejar de compartir un nombre de dominio personalizado privado mediante AWS RAM

Para dejar de compartir su nombre de dominio personalizado privado, primero debe eliminar la asociación del recurso compartido para impedir que el consumidor de API cree más asociaciones de acceso a nombres de dominio. A continuación, debe rechazar la asociación de acceso a nombres

de dominio y eliminar el punto de conexión de VPC de la `policy` del servicio `execute-api`. El consumidor de API puede entonces eliminar su asociación de acceso a nombres de dominio.

Cómo dejar de compartir su nombre de dominio personalizado privado

En primer lugar, debe detener el uso compartido de recursos mediante AWS RAM.

Consola de administración de AWS

Para usar la Consola de administración de AWS, consulte Actualización de un recurso compartido en AWS RAM.

AWS CLI

El siguiente comando disassociate-resource-share permite disociar un recurso compartido del nombre de dominio personalizado privado.

```
aws ram disassociate-resource-share \
    --region us-west-2 \
    --resource-arns arn:aws:apigateway:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234 \
    --principals 222222222222
```

Rechazo de la asociación de acceso a nombres de dominio

Cuando deje de compartir el recurso mediante AWS RAM, debe rechazar la asociación de acceso a nombres de dominio entre un punto de conexión de VPC de otra cuenta y su nombre de dominio personalizado privado.

> **Note**
>
> No puede rechazar una asociación de acceso a nombres de dominio en su propia cuenta. Para dejar de compartir recursos, elimine la asociación de acceso a nombres de dominio. Para obtener más información, consulte Eliminación de una asociación de acceso a nombres de dominio.

Cuando rechaza una asociación de acceso a nombres de dominio con un punto de conexión de VPC, si un consumidor de API intenta llamar a su nombre de dominio personalizado privado, API Gateway rechaza la llamada y devuelve un código de estado `403`.

Consola de administración de AWS

Para rechazar una asociación de acceso a nombres de dominio

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Nombres de dominio personalizados.
3. Elija el nombre de dominio personalizado privado que compartió con otras Cuentas de AWS.
4. En Compartir recursos, elija la asociación de acceso a nombres de dominio que desea rechazar.
5. Elija Rechazar asociación.
6. Confirme la selección y, a continuación, elija Rechazar.

AWS CLI

El siguiente comando `reject-domain-name-access-association` rechaza la asociación de acceso a nombres de dominio entre el punto de conexión de la VPC y su nombre de dominio personalizado privado:

```
aws apigateway reject-domain-name-access-association \
    --domain-name-access-association-arn arn:aws:apigateway:us-west-2:444455556666:/
domainnameaccessassociations/domainname/private.example.com+abcd1234/vpcesource/
vpce-abcd1234efg \
    --domain-name-arn arn:aws:apigateway:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234
```

Denegación del acceso al proveedor de API para invocar su nombre de dominio personalizado privado

Tras rechazar la asociación de acceso a nombres de dominio, debe eliminar el punto de conexión de VPC del consumidor de API de la `policy` para el servicio `execute-api`.

Consola de administración de AWS

Para eliminar el punto de conexión de VPC del consumidor de API de la política de recursos

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Nombres de dominio personalizados.
3. Elija el nombre de dominio personalizado privado que compartió con otras Cuentas de AWS.

4. En la pestaña Política de recursos, elija Editar.
5. Elimine el punto de conexión de VPC de la política.
6. Seleccione Save changes (Guardar cambios).

AWS CLI

El siguiente comando update-domain-name utiliza una operación de parche para actualizar la `policy` del servicio `execute-api` para un nombre de dominio personalizado privado. Esta nueva `policy` elimina un ID de punto de conexión de VPC adicional agregado en the section called “Permiso para que otras cuentas invoquen su nombre de dominio personalizado privado”:

```
aws apigateway update-domain-name
    --domain-name private.example.com \
    --domain-name-id abcd1234 \
    --patch-operations op=replace,path=/policy,value='"{\"Version\": \"2012-10-17\",
       \"Statement\": [{\"Effect\": \"Allow\",\"Principal\": \"*\",\"Action\":
 \"execute-api:Invoke\",\"Resource\":[\"execute-api:/*\"]},{\"Effect\": \"Deny\",
\"Principal\": \"*\",\"Action\": \"execute-api:Invoke\",\"Resource\":[\"execute-
api:/*\"],\"Condition\":{\"StringNotEquals\":{\"aws:SourceVpce\": \"vpce-abcd1234efg
\"}}}]}"
```

A continuación, el consumidor de API debe eliminar la asociación de acceso a nombres de dominio. Usted no puede eliminarla en su nombre. Para obtener más información, consulte the section called “Consumidor de API: eliminar su asociación de acceso a nombres de dominio con un nombre de dominio personalizado privado”.

Proveedor de API: compartir su nombre de dominio personalizado privado mediante la de API Gateway AWS CLI

Puede compartir un nombre de dominio personalizado privado mediante la AWS CLI de API Gateway, pero le recomendamos que utilice AWS RAM para reducir la sobrecarga operativa. Para obtener instrucciones sobre cómo usar AWS RAM para compartir su nombre de dominio personalizado privado, consulte the section called “Proveedor de API: compartir su nombre de dominio personalizado privado mediante AWS RAM”.

Para compartir un nombre de dominio personalizado privado mediante la AWS CLI de API Gateway, debe conceder a otra Cuentas de AWS acceso para crear asociaciones de acceso a nombres de dominio e invocar su nombre de dominio personalizado privado. Para ello, actualice la

`managementPolicy` del servicio API Gateway Management y la `policy` del servicio `execute-api` de su nombre de dominio personalizado privado. También debe conceder acceso al punto de conexión de VPC del consumidor de API en la política de recursos para cualquier API privada asignada a su nombre de dominio personalizado privado.

El consumidor de API aún deberá crear una asociación de acceso a nombres de dominio en su propia cuenta, entre su punto de conexión de VPC y su nombre de dominio personalizado privado. Usted no puede hacerlo en su nombre.

Concesión de acceso a su nombre de dominio personalizado privado

Para conceder acceso a su nombre de dominio personalizado privado

1. Para actualizar la `managementPolicy` del servicio API Gateway Management, cree un archivo JSON que contenga la operación de parche para actualizar la política. El siguiente `patch-managementPolicy.json` sustituye la actual `managementPolicy` por una política de ejemplo que concede a las Cuentas de AWS 111122223333 y 444455556666 acceso para crear asociaciones de acceso a nombres de dominio con el nombre de dominio personalizado privado `private.example.com`.

   ```
   [{
       "op": "replace",
       "path": "/managementPolicy",
       "value": "{\"Version\":\"2012-10-17\",       \"Statement\":[{\"Effect
   \":\"Allow\",\"Principal\":{\"AWS\":[\"arn:aws:iam::111122223333:root
   \", \"arn:aws:iam::444455556666:root\"]},\"Action\":
   \"apigateway:CreateAccessAssociation\",\"Resource\":\"arn:aws:apigateway:us-
   west-2:111122223333:/domainnames/private.example.com+abcd1234\"}]}"
   }]
   ```

   El siguiente comando update-domain-name permite actualizar la `managementPolicy` mediante `patch-managementPolicy.json`.

   ```
   aws apigateway update-domain-name \
       --domain-name private.example.com \
       --domain-name-id abcd1234 \
       --patch-operations file://patch-managementPolicy.json
   ```

Una vez que conceda el acceso, debe notificar al consumidor de API que puede formar la asociación de acceso a nombres de dominio. Si utiliza AWS RAM, AWS RAM llevará a cabo este paso en su nombre.

2. Para actualizar la `policy` del servicio `execute-api`, debe crear un archivo JSON que contenga la operación de parche para actualizar la política. El siguiente `patch-policy.json` sustituye la actual `policy` por una política de ejemplo que concede acceso a dos puntos de conexión de VPC para invocar el nombre de dominio personalizado privado `private.example.com`.

```
[{
    "op": "replace",
    "path": "/policy",
    "value": "{\"Version\": \"2012-10-17\",       \"Statement\": [{\"Effect\":
 \"Allow\",\"Principal\": \"*\",\"Action\": \"execute-api:Invoke\",\"Resource\":
 \"arn:aws:execute-api:us-west-2:111122223333:/domainnames/private.example.com
+abcd1234\"},{\"Effect\": \"Deny\",\"Principal\": \"*\",\"Action\": \"execute-
api:Invoke\",\"Resource\": \"arn:aws:execute-api:us-west-2:111122223333:/
domainnames/private.example.com+abcd1234\",\"Condition\": {\"StringNotEquals\":
 {\"aws:SourceVpce\": [\"vpce-abcd1234\",\"vpce-xyzz0000\"]}}}]}"
}]
```

Utilice el siguiente comando update-domain-name para actualizar la `policy` mediante `patch-policy.json`.

```
aws apigateway update-domain-name \
    --domain-name private.example.com \
    --domain-name-id abcd1234 \
    --patch-operations file://patch-policy.json
```

Denegación del acceso a su nombre de dominio personalizado privado

Para dejar de compartir su nombre de dominio personalizado privado, debe rechazar la asociación de acceso a nombres de dominio entre su nombre de dominio personalizado privado y el punto de conexión de VPC del consumidor de API.

Para denegar el acceso a su nombre de dominio personalizado privado

1. El siguiente comando `reject-domain-name-access-association` rechaza la asociación de acceso a nombres de dominio.

```
aws apigateway reject-domain-name-access-association \
    --domain-name-access-association-arn arn:aws:apigateway:us-
west-2:444455556666:/domainnameaccessassociations/domainname/private.example.com
+abcd1234/vpcesource/vpce-abcd1234 \
    --domain-name-arn arn:aws:apigateway:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234
```

2. Modifique el `patch-managementPolicy.json` para eliminar el acceso a la cuenta del proveedor de API a fin de crear una asociación de acceso a nombres de dominio con su nombre de dominio personalizado privado. El siguiente `patch-managementPolicy.json` elimina una cuenta de la `managementPolicy`:

```
[{
     "op": "replace",
     "path": "/managementPolicy",
     "value": "{\"Version\":\"2012-10-17\",       \"Statement\":[{\"Effect\":
\"Allow\",\"Principal\":\"*\",\"Action\":\"apigateway:CreateAccessAssociation
\",\"Resource\":\"arn:aws:apigateway:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234\"}]}"
}]
```

   El siguiente comando update-domain-name permite actualizar la `managementPolicy` mediante `patch-managementPolicy.json`.

```
aws apigateway update-domain-name \
    --domain-name private.example.com \
    --domain-name-id abcd1234 \
    --patch-operations file://patch-managementPolicy.json
```

3. Modifique el `patch-policy.json` para eliminar el acceso del punto de conexión de VPC del proveedor de API para invocar su nombre de dominio personalizado privado. El siguiente `patch-policy.json` elimina el ID del punto de conexión de VPC de la `policy`:

```
[{
    "op": "replace",
    "path": "/policy",
```

```
    "value": "{\"Version\":\"2012-10-17\",       \"Statement\":[{\"Effect\":
\"Allow\",\"Principal\":\"*\",\"Action\":\"execute-api:Invoke\",\"Resource\":
\"arn:aws:execute-api:us-west-2:111122223333:/domainnames/private.example.com
+abcd1234\"},{\"Effect\":\"Deny\",\"Principal\":\"*\",\"Action\":\"execute-
api:Invoke\",\"Resource\":\"arn:aws:execute-api:us-west-2:111122223333:/
domainnames/private.example.com+abcd1234\",\"Condition\":{\"StringNotEquals\":
{\"aws:SourceVpce\":\"vpce-abcd1234\"}}}]}"
}]
```

El siguiente comando update-domain-name permite actualizar la `policy` mediante `patch-policy.json`.

```
aws apigateway update-domain-name \
    --domain-name private.example.com \
    --domain-name-id abcd1234 \
    --patch-operations file://patch-policy.json
```

Políticas de ejemplo utilizadas en este procedimiento

En la siguiente sección se muestran las políticas de ejemplo que se utilizan en el procedimiento anterior.

La siguiente política de ejemplo corresponde a la `managementPolicy` del servicio Amazon API Gateway Management. Esta política concede a las Cuentas de AWS 111122223333 y 444455556666 acceso para crear asociaciones de acceso a nombres de dominio con el nombre de dominio personalizado privado `private.example.com`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "111122223333",
                    "444455556666"
                ]
            },
```

```
            "Action": "apigateway:CreateAccessAssociation",
            "Resource": "arn:aws:apigateway:us-west-2:111122223333:/domainnames/
private.example.com+a1b2c3"
        }
    ]
}
```

La siguiente política de ejemplo corresponde a la `policy` del servicio `execute-api`. Esta política concede a los puntos de conexión de VPC `vpce-abcd1234` y `vpce-xyzz0000` acceso para invocar el nombre de dominio personalizado privado.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "arn:aws:execute-api:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234"
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": "arn:aws:execute-api:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234",
            "Condition": {
                "StringNotEquals": {
                    "aws:SourceVpce": [
                        "vpce-abcd1234",
                        "vpce-xyzz0000"
                    ]
                }
            }
        }
    ]
}
```

Consumidor de API: asociar su punto de conexión de VPC a un nombre de dominio personalizado privado que se comparta con usted

El siguiente procedimiento muestra cómo consumir un nombre de dominio privado en otra Cuenta de AWS. Dependiendo de su relación de confianza con el proveedor de API, AWS RAM podría completar algunas de las tareas en su nombre.

Si se encuentra en una Cuenta de AWS distinta del nombre de dominio personalizado privado, solo puede asociar su punto de conexión de VPC a un nombre de dominio personalizado privado e invocarlo. No puede ver la `policy` ni ningún otro parámetro del nombre de dominio personalizado privado.

Requisitos previos

Se aplican los siguientes requisitos previos para consumir un nombre de dominio personalizado privado en otra Cuenta de AWS:

- Una VPC y un punto de conexión de VPC para el servicio `execute-api`. La VPC debe tener `enableDnsHostnames` y `enableDnsSupport` establecidos en `true`.
- Le recomendamos que configure al menos dos zonas de disponibilidad por punto de conexión de VPC.

(Opcional) Aceptación del recurso compartido de dominio personalizado privado

Si su proveedor de API utilizó AWS RAM para crear un recurso compartido, dispone de 12 horas para aceptarlo. Si pertenece a la misma organización que utiliza AWS Organizations como proveedor de API, el recurso compartido se acepta automáticamente. Si pertenece a una organización que tiene habilitados los recursos compartidos automáticos, el recurso se comparte automáticamente con usted.

Consola de administración de AWS

Para usar la Consola de administración de AWS, consulte Aceptar y rechazar invitaciones a recursos compartidos en la Guía del usuario de AWS RAM.

AWS CLI

Para buscar todos los recursos que se comparten con usted, utilice el siguiente comando get-resource-share-invitations:

```
aws ram get-resource-share-invitations \
```

```
    --region us-west-2
```

Utilice el ARN del recurso compartido resultante para aceptar la invitación al recurso compartido. El siguiente comando accept-resource-share-invitation permite aceptar el recurso compartido.

```
aws ram accept-resource-share-invitation \
    --resource-share-invitation-arn arn:aws:ram:us-west-2:123456789012:resource-
share-invitation/1e3477be-4a95-46b4-bbe0-c4001EXAMPLE \
    --region us-west-2
```

Asociación de su punto de conexión de VPC a un nombre de dominio personalizado privado compartido

Puesto que los nombres de dominio personalizados privados no son exclusivos, debe asociar su punto de conexión de VPC al ARN de nombre de dominio personalizado exclusivo. Después de crear la asociación de acceso a nombres de dominio, el punto de conexión de VPC puede tardar hasta 15 minutos en invocar correctamente el nombre de dominio personalizado privado. Si tiene un punto de conexión de VPC que utiliza para acceder a un nombre de dominio personalizado público, no lo utilice para crear ninguna asociación de acceso a nombres de dominio.

Consola de administración de AWS

Para asociar el punto de conexión de VPC a un nombre de dominio personalizado privado compartido

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Asociaciones de acceso a nombres de dominio.
3. Elija Crear asociación de acceso a nombres de dominio.
4. En ARN del nombre de dominio, seleccione el ARN del nombre de dominio que el proveedor de API compartió con usted.

   Es posible que el ARN del nombre de dominio no aparezca en la lista desplegable. Puede usar la consola de AWS RAM para ver los nombres de dominio que se han compartido con usted y, a continuación, copiar el ARN del nombre de dominio e ingresarlo en este campo.
5. En ID del punto de conexión de VPC, seleccione el ID del punto de conexión de VPC con el que desea formar la asociación de acceso a nombres de dominio.
6. Elija Crear asociación de acceso a nombres de dominio.

AWS CLI

Puesto que los nombres de dominio personalizados privados no son exclusivos, debe asociar su punto de conexión de VPC al ARN de nombre de dominio personalizado exclusivo. Para buscar el ARN del nombre de dominio, utilice uno de los comandos siguientes.

1. AWS RAM

   El siguiente comando list-resources permite obtener una lista de los recursos que se han compartido con usted. Para usar este comando, el proveedor de API debe haber utilizado AWS RAM para compartir su dominio personalizado privado con usted.

   ```
   aws ram list-resources \
       --resource-owner OTHER-ACCOUNTS \
       --region us-west-2
       --resource-type apigateway:Domainnames
   ```

   API Gateway

   El siguiente comando `get-domain-names` permite obtener una lista de todos los nombres de dominio personalizados privados que pertenecen a otras Cuentas de AWS y con los que puede formar asociaciones de acceso a nombres de dominio.

   ```
   aws apigateway get-domain-names \
       --resource-owner OTHER_ACCOUNTS \
       --region us-west-2
   ```

2. Tras recuperar el ARN, utilice API Gateway para crear la asociación de acceso a nombres de dominio entre el punto de conexión de VPC y un nombre de dominio personalizado privado compartido. Utilice el siguiente comando `create-domain-name-access-association`:

   ```
   aws apigateway create-domain-name-access-association \
       --access-association-source-type VPCE \
       --access-association-source 'vpce-1a2b3c4d5e6f1a2b3' \
       --domain-name-arn arn:aws:apigateway:us-west-2:111122223333:/domainnames/
   private.example.com+abcd1234"
   ```

   El resultado será similar al siguiente.

   ```
   {
   ```

```
    "domainNameAccessAssociationARN": "arn:aws:apigateway:us-
west-2:444455556666:/domainnameaccessassociations/domainname/private.example.com
+abcd1234/vpcesource/vpce-abcd1234efg",
    "accessAssociationSource": "vpce-1a2b3c4d5e6f1a2b3",
    "accessAssociationSourceType": "VPCE",
    "domainNameARN" : "arn:aws:apigateway:us-west-1:111122223333:/domainnames/
private.example.com+a1b2c3"
}
```

Una vez que haya asociado el punto de conexión de VPC al nombre de dominio personalizado privado, confirme que el proveedor de API ha actualizado la política de su nombre de dominio personalizado privado para permitir que el punto de conexión de VPC invoque su nombre de dominio. Para obtener más información, consulte the section called “Permiso para que otras cuentas invoquen su nombre de dominio personalizado privado”.

Creación de una zona alojada en Route 53

Para resolver el nombre de dominio personalizado privado, debe crear una zona alojada privada en Route 53. Una zona alojada es un contenedor que incluye información acerca de cómo se desea dirigir el tráfico de un dominio dentro de una o varias VPC sin tener que exponer sus recursos en Internet. Para obtener más información, consulte Uso de zonas alojadas privadas.

Consola de administración de AWS

Para usar la Consola de administración de AWS, consulte Creación de una zona alojada privada en la Guía para desarrolladores de Amazon Route 53.

En Nombre, escriba el nombre del nombre de dominio personalizado privado. En ID de VPC, utilice la VPC que contiene el punto de conexión de VPC que se utilizó para la asociación de acceso a nombres de dominio.

AWS CLI

El siguiente comando create-hosted-zone permite crear una zona alojada privada:

```
aws route53 create-hosted-zone --name private.example.com \
    --caller-reference 2014-04-01-18:47 \
    --hosted-zone-config Comment="command-line version",PrivateZone=true \
    --vpc VPCRegion=us-west-2,VPCId=vpc-abcd1234
```

El resultado contiene el ID de la zona alojada. El ID de la zona alojada se utiliza en los pasos siguientes.

Creación de un registro A de Route 53

Después de crear la zona alojada, debe crear un registro para resolver el dominio personalizado privado. En este ejemplo, se crea un tipo de registro A. Si está utilizando IPv6 para el punto de conexión de VPC, cree un tipo de registro AAAA. Si utiliza la pila doble para el punto de conexión de VPC, cree un tipo de registro AAAA y un tipo de registro A.

Consola de administración de AWS

Para usar la Consola de administración de AWS, consulte Enrutamiento del tráfico a una API de Amazon API Gateway mediante su nombre de dominio.

Utilice Creación rápida y active la opción Alias. Para el punto de conexión, utilice el nombre de DNS del punto de conexión de VPC.

AWS CLI

Para configurar sus registros de DNS para asignar el nombre de dominio personalizado privado al nombre de host del ID de zona alojada designado, cree un archivo JSON que contenga la configuración a fin de establecer un registro de DNS para el nombre de dominio privado.

El siguiente `setup-dns-record.json` muestra cómo crear un registro `A` de DNS para asignar un nombre de dominio personalizado privado a su nombre de host privado. Debe proporcionar el `DNSName` de su ID de DNS de VPC y el ID de la zona alojada que creó en el paso anterior.

```
{
  "Changes": [
    {
      "Action": "UPSERT",
      "ResourceRecordSet": {
        "Name": "private.example.com",
        "Type": "A",
        "AliasTarget": {
          "DNSName": "vpce-abcd1234.execute-api.us-west-2.vpce.amazonaws.com",
          "HostedZoneId": "Z2OJLYMUO9EFXC",
          "EvaluateTargetHealth": false
        }
      }
    }
```

```
  ]
}
```

El siguiente comando change-resource-record-sets permite crear un registro de DNS para su nombre de dominio personalizado privado:

```
aws route53 change-resource-record-sets \
    --hosted-zone-id ZABCDEFG1234 \
    --change-batch file://path/to/your/setup-dns-record.json
```

Reemplace `hosted-zone-id` por el ID de zona alojada de Route 53 del conjunto de registros de DNS de la cuenta. El valor del parámetro `change-batch` apunta al archivo JSON.

Próximos pasos para un consumidor de API

Ahora puede invocar la API privada en su propia Cuenta de AWS. En su VPC, puede utilizar el siguiente comando curl para acceder a su nombre de dominio personalizado privado.

```
curl https://private.example.com/v1
```

Para obtener más información sobre otras formas de invocar sus API privadas, consulte the section called “Invocación de una API privada con un nombre de dominio personalizado”.

Consumidor de API: eliminar su asociación de acceso a nombres de dominio con un nombre de dominio personalizado privado

Si es un consumidor de API, puede eliminar el recurso de asociación de acceso en cualquier momento. El proveedor de API no puede eliminar la asociación de acceso a nombres de dominio en su nombre.

Le recomendamos que elimine siempre las asociaciones de acceso a nombres de dominio que ya no la utilice.

Consola de administración de AWS

Para eliminar la asociación de acceso a nombres de dominio

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Asociaciones de acceso a nombres de dominio.
3. Seleccione la asociación de acceso a nombres de dominio y, a continuación, elija Eliminar.

4. Confirme la selección y, a continuación, elija Eliminar.

AWS CLI

El siguiente comando `delete-access-association` elimina la asociación de acceso:

```
aws apigateway delete-domain-name-access-association \
    --domain-name-access-association-arn 'arn:aws:apigateway:us-
west-2:444455556666:/domainnameaccessassociations/domainname/private.example.com
+abcd1234/vpcesource/vpce-abcd1234efg'
```

Creación de un nombre de dominio personalizado para API privadas mediante CloudFormation

La siguiente plantilla CloudFormation de ejemplo crea una API privada y un nombre de dominio personalizado privado, asigna la API privada al nombre de dominio personalizado y, a continuación, crea una asociación de acceso a nombres de dominio. Debe proporcionar su propio punto de conexión de VPC, nombre de dominio y ARN de certificado.

Las siguientes consideraciones pueden afectar al uso que haga de CloudFormation para crear un nombre de dominio personalizado privado.

- No puede rechazar una asociación de acceso a nombres de dominio mediante CloudFormation. Para rechazar una asociación de acceso a nombres de dominio, utilice la AWS CLI.
- Utilice la propiedad `AWS::ApiGateway::DomainNameV2` de CloudFormation para crear un nombre de dominio personalizado privado.
- Utilice la propiedad `AWS::ApiGateway:BasePathMappingV2` de CloudFormation para crear una asignación de ruta base.

```
AWSTemplateFormatVersion: 2010-09-09
Parameters:
  EndpointID:
    Type: String
    Default: vpce-abcd1234567efg
    Description: A VPC endpoint with enableDnsHostnames and enableDnsSupport set to
 true.
  DomainName:
    Type: String
    Default: private.example.com
```

```
    Description: A domain name that you own.
  CertificateArn:
    Type: String
    Default: arn:aws:acm:us-west-2:123456789:certificate/abcd-000-1234-0000-000000abcd
    Description: An ACM certificate that covers the domain name.
Resources:
  PrivateApi:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      EndpointConfiguration:
        Types:
          - PRIVATE
        VpcEndpointIds:
          - !Ref EndpointID
      Name: private-api
      Policy:
        Statement:
          - Action: 'execute-api:Invoke'
            Effect: Allow
            Principal: '*'
            Resource: 'execute-api:/*'
          - Action: 'execute-api:Invoke'
            Condition:
              StringNotEquals:
                'aws:SourceVpce': !Ref EndpointID
            Effect: Deny
            Principal: '*'
            Resource: 'execute-api:/*'
        Version: 2012-10-17
  PrivateApiDeployment:
    Type: 'AWS::ApiGateway::Deployment'
    Properties:
      RestApiId: !Ref PrivateApi
      Description: Private API deployment
    DependsOn:
      - PrivateApiMethod
  PrivateApiStage:
    Type: 'AWS::ApiGateway::Stage'
    Properties:
      RestApiId: !Ref PrivateApi
      DeploymentId: !Ref PrivateApiDeployment
      StageName: prod
  PrivateApiMethod:
    Type: 'AWS::ApiGateway::Method'
```

```
    Properties:
      HttpMethod: ANY
      ResourceId: !GetAtt PrivateApi.RootResourceId
      RestApiId: !Ref PrivateApi
      AuthorizationType: NONE
      Integration:
        Type: MOCK
        RequestTemplates:
          application/json: "{\"statusCode\": 200}"
        IntegrationResponses:
          - StatusCode: '200'
      MethodResponses:
        - StatusCode: '200'
  PrivateDomainName:
    Type: AWS::ApiGateway::DomainNameV2
    Properties:
      DomainName: !Ref DomainName
      CertificateArn: !Ref CertificateArn
      EndpointConfiguration:
        Types:
          - PRIVATE
      SecurityPolicy: TLS_1_2
      Policy:
        Statement:
            - Action: 'execute-api:Invoke'
              Effect: Allow
              Principal: '*'
              Resource: 'execute-api:/*'
            - Action: 'execute-api:Invoke'
              Condition:
                StringNotEquals:
                  'aws:SourceVpce': !Ref EndpointID
              Effect: Deny
              Principal: '*'
              Resource: 'execute-api:/*'
        Version: 2012-10-17
  PrivateBasePathMapping:
    Type: AWS::ApiGateway::BasePathMappingV2
    DependsOn:
      - PrivateApiStage
    Properties:
      BasePath: prod
      DomainNameArn: !GetAtt PrivateDomainName.DomainNameArn
      RestApiId: !Ref PrivateApi
```

```
      Stage: prod
  DomainNameAccessAssociation:
    Type: AWS::ApiGateway::DomainNameAccessAssociation
    Properties:
      DomainNameArn: !GetAtt PrivateDomainName.DomainNameArn
      AccessAssociationSource: !Ref EndpointID
      AccessAssociationSourceType: VPCE
```

Invocación de una API privada

Solo puede invocar una API privada en una VPC mediante un punto de conexión de VPC. La API privada debe tener una política de recursos que permita que las VPC y los puntos de conexión de VPC específicos invoquen la API.

Si invoca una API privada sin usar un nombre de dominio personalizado o nombres de DNS privados y el nombre de las API o dominio utiliza una política de seguridad que comienza por `SecurityPolicy_`, debe configurar el modo de acceso de punto de conexión a `BASIC`. Para obtener más información, consulte the section called “Modo de acceso de punto de conexión”.

Invocación de una API privada con un nombre de dominio personalizado

Para invocar una API privada con un nombre de dominio personalizado, el punto de conexión de VPC necesita una asociación de acceso a nombre de dominio con un nombre de dominio personalizado, y el nombre de dominio personalizado debe permitir el acceso para que el punto de conexión de VPC lo invoque. Para obtener más información, consulte the section called “Nombres de dominio personalizados para API privadas”.

No existe ninguna diferencia entre invocar un nombre de dominio personalizado privado de una VPC de su propia Cuenta de AWS o de una Cuenta de AWS distinta.

Uso del nombre de dominio personalizado

Dentro de la VPC, puede invocar la API con el nombre de dominio personalizado. El siguiente ejemplo corresponde a un comando curl para invocar el nombre de dominio personalizado privado:

```
curl https://private.example.com
```

Uso de nombres de host DNS privados específicos de un punto de conexión

Puede invocar la API utilizando el nombre de dominio personalizado y el nombre de host DNS privado específico del punto de conexión.

```
curl https://private-dns-hostname.execute-api.region.vpce.amazonaws.com/basepath -H
 'Host:custom-domain-name'
```

El siguiente ejemplo corresponde a un comando curl para invocar el nombre de dominio personalizado utilizando un nombre de host DNS privado específico de un punto de conexión:

```
curl https://vpce-123456-abc000.execute-api.us-east-2.vpce.amazonaws.com/test -H
 'Host:private.example.com'
```

Invocación de una API privada sin utilizar un nombre de dominio personalizado

Para invocar la API privada sin usar un nombre de dominio personalizado, debe identificar los nombres DNS de la API. El siguiente procedimiento muestra cómo buscar los nombres de DNS.

Consola de administración de AWS

Búsqueda de los nombres de DNS

1. Inicie sesión en la Consola de administración de AWS y abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc/.
2. En el panel de navegación principal, elija Puntos de conexión y, a continuación, elija el punto de conexión de VPC de la interfaz para API Gateway.
3. En el panel Detalles, verá cinco valores en el campo Nombres de DNS. Los tres primeros son los nombres de DNS públicos de la API. Los otros dos son los nombres de DNS privados.

AWS CLI

Use el siguiente comando describe-vpc-endpoints para mostrar los valores de DNS.

```
aws ec2 describe-vpc-endpoints --vpc-endpoint-ids vpce-01234567abcdef012
```

Los tres primeros son los nombres de DNS públicos de la API. Los otros dos son los nombres de DNS privados.

Invocación de una API privada a través de un alias de Route53

Puede asociar o desasociar un punto de conexión de VPC con la API privada. Para obtener más información, consulte the section called “(Opcional) Asociación o desasociación de un punto de conexión de VPC con una API privada”.

Tras asociar los puntos de conexión de VPC a la API privada, puede utilizar la siguiente URL base para invocar la API:

```
https://{rest-api-id}-{vpce-id}.execute-api.{region}.amazonaws.com/{stage}
```

Por ejemplo, si configuró el método `GET /pets` para la etapa `test` y el ID de la API de REST era `01234567ab`, el ID de punto de conexión de VPC era `vpce-01234567abcdef012` y la región era `us-west-2`, puede invocar la API de la siguiente manera:

```
curl -v https://01234567ab-vpce-01234567abcdef012.execute-api.us-west-2.amazonaws.com/
test/pets
```

Invocación de una API privada a través de nombres de DNS privados

Si ha habilitado un DNS privado, puede acceder a la API privada con el nombre de DNS privado siguiente:

```
{restapi-id}.execute-api.{region}.amazonaws.com
```

La URL base para invocar la API tiene el formato siguiente:

```
https://{restapi-id}.execute-api.{region}.amazonaws.com/{stage}
```

Por ejemplo, si configuró el método `GET /pets` para la etapa `test` y el ID de la API de REST era `01234567ab` y la región era `us-west-2`, podría invocar la API privada ingresando la siguiente URL en un navegador:

```
https://01234567ab.execute-api.us-west-2.amazonaws.com/test/pets
```

También puede usar el comando cURL siguiente para invocar la API privada:

```
curl -X GET https://01234567ab.execute-api.us-west-2.amazonaws.com/test/pets
```

> **Warning**
>
> Si habilita el DNS privado para el punto de conexión de VPC, no podrá acceder al punto de conexión predeterminado para las API públicas. Para obtener más información, consulte el artículo sobre el tema ¿Por qué no puedo conectarme a mi API pública desde un punto de conexión de VPC de API Gateway?

Invocación de una API privada con Direct Connect

Puede utilizar Direct Connect para establecer una conexión privada dedicada entre una red en las instalaciones y Amazon VPC y acceder al punto de conexión de la API privada a través de esa conexión mediante nombres de DNS públicos.

También puede utilizar nombres DNS privados para acceder a su API privada desde una red local mediante la configuración de un punto de conexión entrante de Amazon Route 53 Resolver y el reenvío de todas las consultas DNS del DNS privado desde su red remota. Para obtener más información, consulte Reenvío de consultas DNS entrantes a sus VPC en la Guía para desarrolladores de Amazon Route 53.

Invocación de la API privada mediante nombres de host de DNS públicos específicos de punto de conexión

Puede obtener acceso a su API privada utilizando nombres de host DNS específicos de punto de conexión. Se trata de nombres de host DNS públicos que contienen el ID de punto de conexión de VPC o el ID de su API privada.

La URL base generada tiene el siguiente formato:

```
https://{public-dns-hostname}.execute-api.{region}.vpce.amazonaws.com/{stage}
```

Por ejemplo, si configura el método `GET /pets` para la etapa `test` y el ID de la API de REST era `abc1234`, el nombre de host de DNS público era `vpce-def-01234567` y la región era `us-west-2`, podría invocar la API privada con su ID de VPCE utilizando el siguiente encabezado `Host` en un comando cURL:

```
curl -v https://vpce-def-01234567.execute-api.us-west-2.vpce.amazonaws.com/test/pets -H
 'Host: abc1234.execute-api.us-west-2.amazonaws.com'
```

También puede invocar la API privada a través de su ID de API mediante el encabezado `x-apigw-api-id` en un comando cURL en el siguiente formato:

```
curl -v https://{public-dns-hostname}.execute-api.{region}.vpce.amazonaws.com/{stage} -H 'x-apigw-api-id:{api-id}'
```

Supervisión de las API de REST en API Gateway

En esta sección, puede obtener información sobre cómo monitorear la API con las métricas de CloudWatch, CloudWatch Logs, Firehose y AWS X-Ray. Al combinar registros de ejecución de CloudWatch y métricas de CloudWatch, puede registrar errores y seguimientos de ejecución, y monitorear el rendimiento de la API. Es posible que también desee registrar las llamadas a la API en Firehose. También puede usar AWS X-Ray para rastrear llamadas a través de los servicios downstream que componen su API.

> **Note**
>
> API Gateway podría no generar registros y métricas en los siguientes casos:
>
> - Errores 413: Entidad de solicitud demasiado grande
> - Errores 431: Campos del encabezado de solicitud demasiado grandes
> - Errores 429: Demasiadas Solicitudes
> - Errores de la serie 400 de solicitudes enviadas a un dominio personalizado que no tiene asignación de API
> - Errores de la serie 500 causados por errores internos
>
> API Gateway no generará registros ni métricas al probar un método de API de REST. Las entradas de CloudWatch son simuladas. Para obtener más información, consulte the section called “Uso de la consola para probar un método de la API de REST”.

Temas

- Supervisión de la ejecución de la API de REST con métricas de Amazon CloudWatch
- Configuración del registro de CloudWatch para las API de REST en API Gateway
- Rastreo de las solicitudes de usuario a las API de REST mediante X-Ray en API Gateway

Supervisión de la ejecución de la API de REST con métricas de Amazon CloudWatch

Puede monitorear la ejecución de la API mediante CloudWatch, que recopila y procesa datos sin formato de API Gateway en métricas legibles casi en tiempo real. Estas estadísticas se registran durante un periodo de 15 meses, para que pueda obtener acceso a información de historial y obtener una mejor perspectiva acerca del desempeño de su aplicación web o servicio. De forma predeterminada, los datos de las métricas de API Gateway se envían automáticamente a CloudWatch en períodos de un minuto. Para obtener más información, consulte ¿Qué es Amazon CloudWatch? en la guía del usuario de Amazon Cloudwatch.

Las métricas mostradas por API Gateway proporcionan información que puede analizar de diferentes maneras. La lista siguiente muestra algunos usos comunes de las métricas que son sugerencias para empezar:

- Monitoree las métricas de IntegrationLatency para medir la capacidad de respuesta del backend.
- Monitoree las métricas de Latency (Latencia) para medir la capacidad de respuesta general de las llamadas a la API.
- Monitoree las métricas de CacheHitCount y CacheMissCount para optimizar las capacidades de caché para conseguir el rendimiento deseado.

Temas

- Dimensiones y métricas de Amazon API Gateway
- Visualización de las métricas de CloudWatch con el panel de API en API Gateway
- Visualización de métricas de API Gateway en la consola de CloudWatch
- Visualización de eventos de registro de API Gateway en la consola de CloudWatch
- Herramientas de supervisión en AWS para API Gateway

Dimensiones y métricas de Amazon API Gateway

A continuación se enumeran las métricas y las dimensiones que API Gateway envía a Amazon CloudWatch. Para obtener más información, consulte Supervisión de la ejecución de la API de REST con métricas de Amazon CloudWatch.

Métricas de API Gateway

Amazon API Gateway envía datos de las métricas a CloudWatch cada minuto.

El espacio de nombres de `AWS/ApiGateway` incluye las siguientes métricas.

Métrica	Descripción
4XXError	El número de errores del cliente capturados en un periodo determinado. API Gateway cuenta los códigos de estado de respuesta de la puerta de enlace modificados como errores 4XXError. La estadística `Sum` representa esta métrica, es decir, el número total de errores 4XXError en el periodo de tiempo especificado. La estadística `Average` representa la tasa de errores 4XXError, es decir, el número total de errores 4XXError dividido entre el número total de solicitudes recibidas durante el periodo de tiempo. El denominador corresponde a la métrica Count (consulte más adelante). Unit: Count
5XXError	El número de errores del servidor capturados en un periodo determinado. La estadística `Sum` representa esta métrica, es decir, el número total de errores 5XXError en el periodo de tiempo especificado. La estadística `Average` representa la tasa de errores 5XXError, es decir, el número total de errores 5XXError dividido entre el número total de solicitudes recibidas durante el periodo de tiempo. El denominador corresponde a la métrica Count (consulte más adelante). Unit: Count

Métrica	Descripción
CacheHitCount	El número de solicitudes atendidas desde la caché de la API en un periodo de tiempo determinado. La estadística `Sum` representa esta métrica, es decir, el número total de aciertos de caché en el periodo de tiempo indicado. La estadística `Average` represent a la tasa de aciertos de caché, es decir, el número total de aciertos de caché dividido por el número total de solicitudes recibidas durante el periodo de tiempo. El denominador corresponde a la métrica Count (consulte más adelante). Unit: Count
CacheMissCount	El número de solicitudes atendidas desde el backend en un periodo determinado, cuando el almacenam iento en caché de la API está habilitado. La estadística `Sum` representa esta métrica, es decir, el número total de errores de caché en el periodo de tiempo indicado. La estadística `Average` represent a la tasa de errores de caché, es decir, el número total de errores de caché dividido entre el número total de solicitudes recibidas durante el periodo de tiempo. El denominador corresponde a la métrica Count (consulte más adelante). Unit: Count
Count	El número total de solicitudes de API en un periodo de tiempo determinado. La estadística `SampleCount` representa esta métrica. Unit: Count

Métrica	Descripción
IntegrationLatency	Es el tiempo entre que API Gateway transmite una solicitud al backend y se recibe una respuesta del backend. Unit: Millisecond
Latency	El tiempo entre que API Gateway recibe una solicitud de un cliente y devuelve una respuesta al cliente. La latencia incluye la latencia de la integración y otra carga de API Gateway. Unit: Millisecond

Dimensiones de las métricas de

Puede usar las dimensiones de la tabla siguiente para filtrar las métricas de API Gateway.

> **Note**
>
> API Gateway elimina los caracteres no ASCII de la dimensión ApiName antes de enviar las métricas a CloudWatch. Si APIName no contiene caracteres ASCII, se utiliza API ID como ApiName.

Dimensión	Descripción
ApiName	Filtra las métricas de API Gateway de la API de REST con el nombre de la API especificado.
ApiName, Method, Resource, Stage	Filtra las métricas de API Gateway del método de la API con el nombre de la API, la etapa, el recurso y el método especificados. API Gateway no enviará estas métricas a menos que haya habilitado explícitamente las métricas detallada s de CloudWatch. En la consola, elija una etapa y, a

Dimensión	Descripción
	continuación, en Registros y seguimiento, seleccione Editar. Seleccione Métricas detalladas y, a continuación, seleccione Guardar cambios. También puede llamar al comando update-stage AWS CLI para actualizar la propiedad `metricsEnabled` a `true`. Si activa estas métricas, se le cobrarán cargos adicionales en su cuenta. Para obtener más información sobre precios, consulte Precios de Amazon CloudWatch.
ApiName, Stage	Filtra las métricas de API Gateway para el recurso de la etapa de la API con el nombre y la etapa de la API especificados.

Visualización de las métricas de CloudWatch con el panel de API en API Gateway

Puede utilizar el panel de API de la consola de API Gateway para mostrar las métricas de CloudWatch de la API implementada en API Gateway. Estas se muestran como un resumen de la actividad de la API a lo largo del tiempo.

Temas

- Requisitos previos
- Examinar actividades de la API en el panel

Requisitos previos

1. Debe tener una API creada en API Gateway. Siga las instrucciones en Desarrollo de las API de REST en API Gateway.
2. Debe haber implementado la API al menos una vez. Siga las instrucciones en Implementación de las API de REST en API Gateway.

Examinar actividades de la API en el panel

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.

2. Elegir una API.
3. En el panel de navegación principal, elija Panel.
4. En Etapa, elija la etapa deseada.
5. Elija Rango de fechas para especificar un rango de fechas.
6. Actualice, si es necesario, y consulte las métricas individuales que se muestran en gráficos independientes denominados Llamadas a la API, Latencia, Latencia de integración, Latencia, Error 4xx y Error 5xx.

> **Tip**
>
> Para examinar las métricas de CloudWatch de nivel de método, asegúrese de haber habilitado CloudWatch Logs en un nivel de método. Para obtener más información sobre cómo configurar el registro en el nivel de método, consulte Anule la configuración a nivel de etapa.

Visualización de métricas de API Gateway en la consola de CloudWatch

Las métricas se agrupan en primer lugar por el espacio de nombres de servicio y, a continuación, por las diversas combinaciones de dimensiones dentro de cada espacio de nombres. Para ver las métricas en el nivel de método de la API, active las métricas detalladas. Para obtener más información, consulte the section called “Actualización de la configuración de etapas”.

Para ver las métricas de API Gateway mediante la consola de CloudWatch

1. Abra la consola de CloudWatch en https://console.aws.amazon.com/cloudwatch/.
2. Si es necesario, cambie la Región de AWS. En la barra de navegación, seleccione la región donde residen sus recursos de AWS.
3. En el panel de navegación, seleccione Metrics (Métricas).
4. En la pestaña All metrics (Todas las métricas), seleccione API Gateway.
5. Para ver las métricas por etapa, elija el panel By Stage (Por etapa). A continuación, seleccione los nombres de las métricas y las API.
6. Para ver las métricas por API específica, elija el panel By Api Name (Por nombre de API). A continuación, seleccione los nombres de las métricas y las API.

Para ver las métricas mediante la AWS CLI

1. Utilice el siguiente comando list-metrics para obtener una lista de métricas:

   ```
   aws cloudwatch list-metrics --namespace "AWS/ApiGateway"
   ```

   Después de crear una métrica, espere hasta 15 minutos para que aparezca la métrica. Para ver las estadísticas de las métricas con mayor rapidez, use get-metric-data o get-metric-statistics.
2. Utilice el siguiente comando get-metrics-statistics para ver el promedio de un periodo de tiempo a intervalos de 5 minutos:

   ```
   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
   ```

Visualización de eventos de registro de API Gateway en la consola de CloudWatch

En la siguiente sección se explican los requisitos previos necesarios y cómo ver los eventos de registro de API Gateway en la consola de CloudWatch.

Requisitos previos

1. Debe tener una API creada en API Gateway. Siga las instrucciones en Desarrollo de las API de REST en API Gateway.
2. Debe haber implementado e invocado la API al menos una vez. Siga las instrucciones en Implementación de las API de REST en API Gateway y Invocación de las API de REST en API Gateway.
3. Debe tener habilitados los registros de CloudWatch para una etapa. Siga las instrucciones en Configuración del registro de CloudWatch para las API de REST en API Gateway.

Para ver las solicitudes de API registradas y sus respuestas mediante la consola de CloudWatch

1. Abra la consola de CloudWatch en https://console.aws.amazon.com/cloudwatch/.
2. Si es necesario, cambie la Región de AWS. En la barra de navegación, seleccione la región donde residen sus recursos de AWS. Para obtener más información, consulte Regiones y puntos de conexión.
3. En el panel de navegación, elija Logs (Registros), Log groups (Grupos de registro).

4. En la tabla Log Groups (Grupos de registros), elija un grupo de registros con el nombre API-Gateway-Execution-Logs_{rest-api-id}/{stage-name}.
5. En la tabla Log Streams (Flujos de registros), elija un flujo de registros. Puede utilizar la marca de tiempo para ayudar a localizar el flujo de registros de su interés.
6. Elija Text (Texto) para ver el texto sin formato o elija Row (Fila) para ver el evento fila por fila.

> **Important**
>
> CloudWatch le permite eliminar grupos o flujos de registros. No elimine manualmente los grupos o flujos de registros de API de API Gateway; deje que API Gateway administre estos recursos. Si los grupos o flujos de logs se eliminan manualmente, es posible que no se registren las solicitudes de API ni las respuestas. Si esto ocurre, puede eliminar todo el grupo de registros para la API y volver a implementar la API. Esto se debe a que API Gateway crea grupos de registros o flujos de registros para una etapa de API en el momento en que se implementa.

Herramientas de supervisión en AWS para API Gateway

AWS proporciona varias herramientas que puede utilizar para monitorear API Gateway. Puede configurar algunas de estas herramientas para que monitoricen automáticamente, pero otras herramientas requieren intervención manual. Le recomendamos que automatice las tareas de monitorización en la medida de lo posible.

Herramientas de monitoreo automatizadas en AWS

Puede utilizar las siguientes herramientas de monitoreo automatizado para vigilar API Gateway e informar cuando haya algún problema:

- Alarmas de Amazon CloudWatch: vigile una métrica durante un periodo de tiempo especificado y realice una o varias acciones según el valor que tenga la métrica en comparación con un determinado umbral durante una serie de periodos de tiempo. La acción es una notificación enviada a un tema de Amazon Simple Notification Service (Amazon SNS) o a una política de Amazon EC2 Auto Scaling. Las alarmas de CloudWatch no invocan acciones tan solo por tener un estado determinado; es necesario que el estado haya cambiado y se mantenga durante un número específico de periodos. Para obtener más información, consulte Supervisión de la ejecución de la API de REST con métricas de Amazon CloudWatch.

- Registros de Amazon CloudWatch: monitorice, almacene y obtenga acceso a los archivos de registro de AWS CloudTrail u otras fuentes. Para obtener más información, consulte ¿Qué es registros de CloudWatch? en la Guía del usuario de Amazon CloudWatch.
- Amazon EventBridge (anteriormente se llamaba CloudWatch Events): seleccione los eventos y diríjalos hacia uno o varios flujos o funciones de destino para realizar cambios, capturar información de estado y aplicar medidas correctivas. Para obtener más información, consulte ¿Qué es Amazon EventBridge? en la Guía del usuario de EventBridge.
- Monitoreo de registros de AWS CloudTrail: comparta archivos de registro entre cuentas, monitoree los archivos de registro de CloudTrail en tiempo real enviándolos a CloudWatch Logs, escriba aplicaciones de procesamiento de registros en Java y compruebe que los archivos de registro no hayan cambiado después de que CloudTrail los entregara. Para obtener más información, consulte Uso de archivos de registro de CloudTrail en la Guía del usuario de AWS CloudTrail.

Herramientas de monitoreo manuales

Otra parte importante del monitoreo de API Gateway implica el monitoreo manual de los elementos que no cubren las alarmas de CloudWatch. Los paneles de API Gateway, CloudWatch y otras consolas de AWS proporcionan una vista rápida del estado del entorno de AWS. Es recomendable que también compruebe los archivos de registro de la ejecución de la API.

- Los paneles de API Gateway muestran las siguientes estadísticas para una etapa de API determinada durante un periodo de tiempo especificado:
  - API Calls (Llamadas a la API
  - Cache Hit (Acierto de caché), solo cuando está habilitado el almacenamiento en caché de la API.
  - Cache Miss (Error de caché), solo cuando está habilitado el almacenamiento en caché de la API.
  - Latency (Latencia
  - Integration Latency (Latencia de integración
  - 4XX Error (Error 4XX
  - 5XX Error (Error 5XX
- La página principal de CloudWatch muestra:
  - Alarmas y estado actual
  - Gráficos de alarmas y recursos
  - Estado de los servicios

Además, puede utilizar CloudWatch para hacer lo siguiente:

- Crear paneles personalizados para monitorizar los servicios que le interesan
- Realizar un gráfico con los datos de las métricas para resolver problemas y descubrir tendencias
- Buscar y examinar todas sus métricas de recursos de AWS.
- Crear y editar las alarmas de notificación de problemas

Creación de alarmas de CloudWatch para monitorear API Gateway

Puede crear una alarma de CloudWatch que envíe un mensaje de Amazon SNS cuando la alarma cambia de estado. Una alarma vigila una única métrica durante el período especificado y realiza una o varias acciones en función del valor de la métrica relativo a un determinado umbral durante una serie de períodos de tiempo. La acción es una notificación que se envía a un tema de Amazon SNS o a una política de escalado automático. Las alarmas invocan acciones únicamente para los cambios de estado prolongados. Las alarmas de CloudWatch no invocan acciones tan solo por tener un estado determinado; es necesario que el estado haya cambiado y se mantenga durante un número específico de periodos.

Configuración del registro de CloudWatch para las API de REST en API Gateway

Para ayudarle a depurar problemas relacionados con la ejecución de solicitudes o el acceso de clientes a la API, puede habilitar Amazon CloudWatch Logs para registrar las llamadas a la API. Para obtener más información acerca de CloudWatch, consulte the section called “Métricas de CloudWatch”.

Existen dos tipos de registro de API en CloudWatch:

- Registro de ejecución: API Gateway registra las acciones realizadas para procesar las solicitudes de API, incluidos los errores y los seguimientos de ejecución. También puede configurar la entrega de Registros de Amazon CloudWatch para enrutar los registros de ejecución a los propios destinos.
- Registros de acceso: se registra quién accedió a la API y cómo lo hizo. Cree su propio grupo de registros, elija un formato de registro y especifique qué variables de `$context` desea incluir.

Puede habilitar el registro de ejecución y el registro de acceso por separado.

Permisos de registros de CloudWatch

Para habilitar CloudWatch Logs, debe conceder permiso a API Gateway para leer y escribir registros en CloudWatch para su cuenta. La política administrada AmazonAPIGatewayPushToCloudWatchLogs cuenta con todos los permisos necesarios.

> **Note**
>
> API Gateway llama a AWS Security Token Service para asumir el rol de IAM, por lo que debe asegurarse de que AWS STS esté habilitado para la región. Para obtener más información, consulte Administración de AWS STS en una región de AWS.

Para conceder estos permisos a la cuenta, cree un rol de IAM con `apigateway.amazonaws.com` como entidad de confianza, asocie a este rol la política anterior y establezca su ARN en la propiedad cloudWatchRoleArn de la cuenta. Debe establecer la propiedad cloudWatchRoleArn por separado para cada región de AWS en la que desee habilitar CloudWatch Logs.

Si recibe un error cuando configure el ARN del rol de IAM, verifique la configuración de la cuenta de AWS Security Token Service para asegurarse de que AWS STS esté habilitado en la región que se está utilizando. Para obtener más información acerca de la habilitación de AWS STS, consulte Administración de AWS STS en una región de AWS en la Guía del usuario de IAM.

Temas

- Registro de ejecución para API de REST
- Registros de acceso para las API de REST

Registro de ejecución para API de REST

El registro de ejecución registra las acciones que API Gateway lleva a cabo para procesar una solicitud de API. Los datos registrados contienen errores, rastros de ejecución, cargas útiles o valores de parámetros de solicitud y respuesta, datos utilizados por autorizadores Lambda, si las claves de API son necesarias y si los planes de uso están habilitados. API Gateway redacta los encabezados de autorización, los valores de las claves de API y otros parámetros de solicitud confidenciales similares a partir de los datos registrados.

Para aumentar la seguridad, le recomendamos que utilice el registro de ejecución en el nivel `INFO` o `ERROR`. Es probable que deba hacerlo para asegurar el cumplimiento de diversos marcos normativos.

Para obtener más información, consulte Controles de Amazon API Gateway en la Guía del usuario de AWS Security Hub.

Registro de ejecución estándar

Al habilitar el registro de ejecución en una etapa, API Gateway crea y administra automáticamente un grupo de registros de Registros de CloudWatch. El nombre del grupo de registro es `API-Gateway-Execution-Logs_{rest-api-id}/{stage_name}`. Dentro de cada grupo de registro, los registros se dividen en flujos de registros ordenados por la hora del último evento.

Con el registro de ejecución estándar:

- Los eventos de registro se truncan a 1 KB.
- API Gateway administra el nombre del grupo de registro. No puede elegir el nombre del grupo de registro ni enviar registros directamente a Amazon S3 o Firehose.

Para habilitar el registro de ejecución estándar, establezca el `loggingLevel` en la etapa como `ERROR` o `INFO`. No se necesitan más configuraciones.

Permisos

Para habilitar CloudWatch Logs, debe conceder permiso a API Gateway para leer y escribir registros en CloudWatch para su cuenta. La política administrada AmazonAPIGatewayPushToCloudWatchLogs cuenta con todos los permisos necesarios.

Cree un rol de IAM con `apigateway.amazonaws.com` como entidad de confianza, asocie la política anterior y establezca el ARN del rol de IAM en la propiedad cloudWatchRoleArn de la cuenta. Debe establecer esta propiedad de forma independiente para cada región de AWS en la que desee habilitar Registros de CloudWatch.

> **Note**
>
> API Gateway llama a AWS Security Token Service para asumir el rol de IAM, por lo que debe asegurarse de que AWS STS esté habilitado para la región. Para obtener más información, consulte Administración de AWS STS en una región de AWS.

Habilitación del registro de ejecución (consola)

Antes de empezar, implemente la API en una etapa y configure un ARN de rol de Registros de CloudWatch adecuado para la cuenta.

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Configuración y, a continuación, en Registro, elija Editar.
3. En ARN de rol de registro de CloudWatch, ingrese el ARN de un rol de IAM con los permisos adecuados. Tendrá que hacer esto una vez para cada una de las Cuenta de AWS que creen API mediante API Gateway.
4. En el panel de navegación principal, elija API, elija la API de REST y, a continuación, elija Etapas.
5. Elija una etapa y, a continuación, en la sección Registros y rastreo, elija Editar.
6. Elija un nivel de registro en el menú desplegable Registros de CloudWatch:
   - Desactivado: API Gateway no registra las solicitudes de esta etapa.
   - Solo errores: API Gateway solo registra los errores.
   - Errores y registros de información: API Gateway registra todos los eventos.
7. (Opcional) Seleccione el seguimiento de datos para registrar todos los datos de solicitud y respuesta. Esto puede dar como resultado el registro de información confidencial.

   > **Note**
   >
   > Le recomendamos que no utilice Rastreo de datos para las API de producción.

8. (Opcional) Seleccione Métricas detalladas para activar las métricas detalladas de CloudWatch. Para obtener más información, consulte the section called “Métricas de CloudWatch”.
9. Elija Save changes (Guardar cambios).

No es necesario volver a implementar la API cuando se actualiza la configuración de la etapa.

Habilitación del registro de ejecución (CloudFormation)

Utilice la propiedad `MethodSettings` del recurso de etapa para habilitar el registro de la ejecución.

```
  TestStage:
```

```
    Type: AWS::ApiGateway::Stage
    Properties:
      StageName: test
      RestApiId: !Ref MyAPI
      DeploymentId: !Ref Deployment
      MethodSettings:
        - ResourcePath: "/*"
          HttpMethod: "*"
          LoggingLevel: INFO
```

Para habilitar CloudWatch Logs, debe conceder permiso a API Gateway para leer y escribir registros en CloudWatch para su cuenta. Para obtener más información, consulte Asociar una cuenta a un rol de IAM en la Guía del usuario de AWS CloudFormation.

Entrega de Registros de Amazon CloudWatch para los registros de ejecución

La entrega de Registros de Amazon CloudWatch proporciona una ruta alternativa para los registros de ejecución. En lugar de escribir en un grupo de registro administrado por API Gateway, configure una entrega que envíe los registros de ejecución a los destinos que usted posee y controla.

Con la entrega de Registros de Amazon CloudWatch:

- Los eventos de registro pueden ocupar hasta 1 MB, lo que permite capturar datos de solicitud y respuesta más completos. Las cargas útiles que superan 1 MB siguen truncadas.
- Se elige el destino: los grupos de registro de Registros de CloudWatch propios, buckets de Amazon S3 o transmisiones de Firehose.
- Puede enviar registros a varios destinos simultáneamente.
- Se controla la configuración del grupo de registro, incluidos los nombres y la retención.

> **Important**
>
> La entrega de Registros de CloudWatch sustituye al registro de ejecución estándar. Al configurar una entrega, API Gateway deja de escribir en el grupo de registro administrado automáticamente. No puede usar ambas rutas al mismo tiempo para la misma etapa.

Al crear una entrega, API Gateway dirige los registros de ejecución a los destinos configurados en lugar de al grupo de registro administrado automáticamente. Al eliminar una entrega, los registros de

ejecución vuelven a fluir automáticamente al grupo de registro administrado automáticamente. No es necesario realizar cambios en la configuración de API Gateway más allá del requisito previo de configurar `loggingLevel` en `ERROR` o `INFO`.

> **Note**
>
> La entrega de registros se realiza bajo un modelo de mejor esfuerzo. En casos excepcionales, es posible que algunos eventos de registro no se entreguen.

> **Warning**
>
> Antes de habilitar la entrega de registros, actualice los paneles, las alarmas o los filtros de suscripción que hagan referencia al grupo de registro administrado automáticamente (`API-Gateway-Execution-Logs_{rest-api-id}/{stage_name}`). Una vez que se activa la entrega, ese grupo de registro ya no recibe nuevos eventos.

En la siguiente tabla, se compara el registro de ejecución estándar con la entrega de Registros de Amazon CloudWatch.

	Registro de ejecución estándar	Entrega de Registros de Amazon CloudWatch
Tamaño máximo del evento de registro	1 KB	1 MB
Destinos	Registros de CloudWatch (grupo de registro administrado)	Registros de CloudWatch, Amazon S3, Firehose (los recursos propios)
Múltiples destinos	No	Sí
Nombre de grupo de registro	Corregido por API Gateway	Se elige
Configuración	automática cuando el registro está habilitado	Requiere crear una entrega de Registros de Amazon CloudWatch

	Registro de ejecución estándar	Entrega de Registros de Amazon CloudWatch
Requisito previo	Establezca `loggingLevel` en `ERROR` o `INFO`	Establezca `loggingLevel` en `ERROR` o `INFO`

> **Note**
>
> Si configura `loggingLevel` en `OFF` después de configurar una entrega, no se generará ni entregará ningún registro de ejecución. Los destinos de entrega permanecen configurados, pero no reciben datos.

Para crear una entrega, consulte the section called “Creación de una entrega de registros”.

Creación de una entrega de registros para los registros de ejecución de la API de REST

Puede enrutar los registros de ejecución a los grupos de registros de Registros de CloudWatch propios, buckets de Amazon S3 o transmisiones de Firehose. Para ello, cree una entrega de Registros de Amazon CloudWatch. Esto implica tres pasos: creación de un origen de entrega, creación de un destino de entrega y vincularlos a una entrega.

Requisitos previos

Antes de crear una entrega de registros, asegúrese de disponer de lo siguiente:

- El registro de ejecución está habilitado en la etapa de la API de REST. Establezca `loggingLevel` en `ERROR` o `INFO`. Si el registro no está habilitado, se produce un error en la creación del origen de entrega.
- Un recurso de destino: un grupo de registro de Registros de CloudWatch, un bucket de Amazon S3 o una transmisión de Firehose de su propiedad.
- Permisos para llamar a las API de entrega de Registros de Amazon CloudWatch. Para obtener más información, consulte Habilitación del registro de servicios de AWS en la Guía del usuario de Registros de Amazon CloudWatch.

Valores específicos de API Gateway para la entrega de Registros de Amazon CloudWatch

Al crear una entrega para los registros de ejecución de API Gateway, utilice los siguientes valores:

ARN de recurso

```
arn:aws:apigateway:{region}:{account-id}:/restapis/{rest-api-id}/
stages/{stage-name}
```

Tipo de registro

```
EXECUTION_LOGS
```

Creación de una entrega de registros (AWS CLI)

En el siguiente ejemplo, se crea una entrega que envía los registros de ejecución desde una API de REST a un grupo de registros de Registros de CloudWatch que sea de su propiedad.

1. Creación del origen de entrega

   Registre la API de REST como origen de entrega.

   ```
   aws logs put-delivery-source \
       --name my-apigw-source \
       --log-type EXECUTION_LOGS \
       --resource-arn arn:aws:apigateway:{region}:{account-id}:/restapis/{rest-api-
   id}/stages/{stage-name}
   ```

2. Creación del destino de la entrega

   Registre el grupo de registros como un destino de entrega.

   ```
   aws logs put-delivery-destination \
       --name my-log-destination \
       --delivery-destination-configuration
    "destinationResourceArn=arn:aws:logs:{region}:{account-id}:log-group:{log-group-
   name}"
   ```

3. Creación de la entrega

   Enlace el origen al destino.

   ```
   aws logs create-delivery \
       --delivery-source-name my-apigw-source \
       --delivery-destination-arn arn:aws:logs:{region}:{account-id}:delivery-
   destination:my-log-destination
   ```

Para enviar registros a Amazon S3 o Firehose, especifique el ARN del recurso correspondiente en el comando `put-delivery-destination`. Para obtener más información acerca del flujo de trabajo de entrega de Registros de Amazon CloudWatch completo, consulte Habilitación de registros de servicios de AWS en la Guía del usuario de Registros de Amazon CloudWatch.

Para enviar registros a varios destinos, cree entregas adicionales con el mismo origen de entrega con destinos diferentes.

Campos de registro

Con la entrega de Registros de Amazon CloudWatch, los registros de ejecución incluyen campos estructurados que puede personalizar por destino. Están disponibles los siguientes campos:

- `resource_arn`: el ARN de la etapa de la API
- `event_timestamp`: la hora del evento de registro
- `api_id`: el identificador de la API de REST
- `stage`: el nombre de la etapa
- `resource_path`: la ruta de recurso de la solicitud
- `http_method`: el método HTTP
- `payload`: el contenido del registro de ejecución (igual que el de los registros de ejecución estándar)

Puede seleccionar los campos que desea incluir y el formato de salida para cada destino mediante el parámetro `--record-fields` en `create-delivery` y el parámetro `--output-format` en `put-delivery-destination`.

Los formatos de salida disponibles dependen del destino:

- Registros de CloudWatch: json, plain
- Amazon S3: json, plain, w3c, parquet
- Firehose: json, plain

> **Coincidencia con el resultado del registro de ejecución estándar**
>
> Si desea que el resultado de la entrega coincida con el registro de ejecución estándar (solo carga útil, sin campos adicionales), agregue `--output-format "plain"` a `put-`

`delivery-destination` y `--record-fields "payload" --field-delimiter ""` a `create-delivery`.

Creación de una entrega de registros (CloudFormation)

Al ensamblar el ARN de la etapa para las plantillas de CloudFormation, use el siguiente patrón:

```
  DeliverySource:
    Type: AWS::Logs::DeliverySource
    Properties:
      Name: my-apigw-source
      LogType: EXECUTION_LOGS
      ResourceArn: !Sub "arn:${AWS::Partition}:apigateway:${AWS::Region}:
${AWS::AccountId}:/restapis/${MyApi}/stages/${MyStageName}"
```

Creación de una entrega de registros (consola de API Gateway)

Puede crear una entrega de registros desde la sección Registros y rastreo de la página de detalles de la etapa.

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Elija la API de REST y, a continuación, elija Etapas.
3. Elegir una etapa.
4. En la sección Registros y rastreo, en Destinos de entrega de registros, elija Agregar destino.
5. Elija un tipo de destino (Registros de CloudWatch, Amazon S3 o Firehose) y especifique el recurso de destino.
6. Seleccione Save.

> **Important**
>
> Debe habilitar el registro de ejecución en la etapa antes de poder agregar destinos de entrega. Si el registro está establecido en Desactivado, la configuración de entrega no genera registros.

> **Note**
>
> Para configurar los destinos de entrega entre cuentas, utilice la AWS CLI. Debe crear una política de destino de entrega en la cuenta de destino antes de crear la entrega en la cuenta de origen. Para obtener más información, consulte Ejemplo de entrega entre cuentas en la Guía del usuario de Registros de Amazon CloudWatch.

Verificación de la entrega de registros de ejecución

Tras crear una entrega de registros, compruebe que la entrega esté activa y que los registros de ejecución lleguen al destino.

Para confirmar que una entrega está activa, use el comando `describe-deliveries`:

```
aws logs describe-deliveries
```

A continuación, envíe una solicitud de prueba y compruebe el destino:

1. Envíe una solicitud de prueba a la API de REST.

   ```
   curl https://{rest-api-id}.execute-api.{region}.amazonaws.com/{stage}/{resource}
   ```

2. Espere 1-2 minutos para que los registros se propaguen.
3. Compruebe el destino para los eventos de registro:

   - Registros de CloudWatch: abra la consola de Registros de CloudWatch y navegue hasta el grupo de registro. Debería ver un flujo de registro con los eventos recientes.
   - Amazon S3: abra la consola de Amazon S3 y vaya al bucket. Busque objetos con un prefijo de ruta que incluya el ID de API de REST.
   - Firehose: compruebe las métricas del flujo de entrega en la consola de Firehose para confirmar que los datos fluyen.

Solución de problemas

Si los registros no llegan al destino, compruebe lo siguiente:

- El registro está habilitado: confirme que `loggingLevel` está configurado en `ERROR` o `INFO` en la etapa. Si el registro está `OFF`, no se generará ningún registro, independientemente de la configuración de entrega.
- Estado de entrega: utilice el comando de la `get-delivery` CLI para comprobar el estado de entrega y confirmar que está activo.

```
aws logs get-delivery --id {delivery-id}
```

- Permisos: compruebe que la política de recursos del destino permite a Registros de Amazon CloudWatch escribir en él. Para obtener más información, consulte Habilitación del registro de servicios de AWS en la Guía del usuario de Registros de Amazon CloudWatch.

Eliminación de una entrega de registros para los registros de ejecución de la API de REST

Al eliminar una entrega de registros, API Gateway vuelve a escribir los registros de ejecución en el grupo de registros administrado automáticamente (`API-Gateway-Execution-Logs_{rest-api-id}/{stage_name}`), siempre y cuando el registro de ejecución siga habilitado en la etapa.

Eliminación de una entrega de registros (AWS CLI)

Para eliminar una entrega, se elimina la entrega y, a continuación, se eliminan de forma opcional el origen y el destino de la entrega.

1. Eliminación de la entrega

```
aws logs delete-delivery --id {delivery-id}
```

2. (Opcional) Eliminación del origen de la entrega

```
aws logs delete-delivery-source --name my-apigw-source
```

3. (Opcional) Eliminación del destino de la entrega

```
aws logs delete-delivery-destination --name my-log-destination
```

> **Note**
>
> La eliminación del origen o el destino de la entrega es opcional. Puede reutilizarlos para crear nuevas entregas más adelante.

Eliminación de una entrega de registros (consola de API Gateway)

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Elija la API de REST y, a continuación, elija Etapas.
3. Elegir una etapa.
4. En la sección Registros y rastreo, en Destinos de entrega de registros, seleccione el destino que desea eliminar.
5. Elija Eliminar.

Mantenimiento de la continuidad del registro

> **Tip**
>
> Para mantener los registros en el mismo grupo de registros y conservar la configuración de retención y los filtros de suscripción existentes, configure el grupo de registros administrado automáticamente (`API-Gateway-Execution-Logs_{rest-api-id}/{stage_name}`) como destino de entrega. Los nombres de los flujos de registro diferirán de los creados por el registro estándar.

Registros de acceso para las API de REST

En los registros de acceso, se registra quién ha accedido a la API y cómo el intermediario ha accedido a la API. Puede crear su propio grupo de registro o elegir un grupo de registro existente. Para especificar los detalles de acceso, se seleccionan variables `$context`, un formato de registro y un destino de grupo de registro.

Formatos de los registro de acceso

El formato del registro de acceso debe incluir al menos `$context.requestId` o `$context.extendedRequestId`. Como práctica recomendada, incluya ambos en el formato de registro.

`$context.requestId`

Esto registra el valor en el encabezado `x-amzn-RequestId`. Los clientes pueden invalidar el valor del encabezado `x-amzn-RequestId` con un valor en formato de identificador único universal (UUID). API Gateway devuelve este ID de solicitud en el encabezado de respuesta `x-amzn-RequestId`. API Gateway sustituye los ID de solicitud invalidados que no tienen el formato de un UUID por *`UUID`*`_REPLACED_INVALID_REQUEST_ID` en los registros de acceso.

`$context.extendedRequestId`

ExtendedRequestID es un ID único que API Gateway genera. API Gateway devuelve este ID de solicitud en el encabezado de respuesta `x-amz-apigw-id`. Una persona que llama a la API no puede proporcionar ni anular este ID de solicitud. Es posible que tenga que proporcionar este valor a AWS Support para que le ayude a solucionar los problemas de la API. Para obtener más información, consulte the section called “Variables para el registro de acceso para API Gateway”.

Elija un formato de registro que también se utilice en el backend de análisis; por ejemplo, Common Log Format (CLF), JSON, XML o CSV. Después, puede incluir datos en los registros de acceso para que tengan las métricas calculadas y procesadas. Para definir el formato del registro, establezca el ARN del grupo de registros en la propiedad accessLogSettings/destinationArn de stage. Puede obtener el ARN de un grupo de registro en la consola de CloudWatch. Para definir el formato del registro de acceso, establezca el formato elegido en la propiedad accessLogSetting/format de stage.

Algunos ejemplos de los formatos de registro de acceso que se utilizan habitualmente se muestran en la consola de API Gateway y se detallan a continuación.

- `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 (valores separados por comas):

```
$context.identity.sourceIp,$context.identity.caller,$context.identity.user,
$context.requestTime,$context.httpMethod,$context.resourcePath,$context.protocol,
$context.status,$context.responseLength,$context.requestId,$context.extendedRequestId
```

Configuración del registro de acceso mediante la consola de API Gateway

Antes de configurar el registro de acceso, implemente la API en una etapa y configure un ARN de rol de Registros de CloudWatch adecuado para la cuenta.

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija API y, a continuación, realice una de las siguientes acciones:
   a. Elija una API existente y después una etapa.
   b. Cree una API e impleméntela como etapa.
3. En el panel de navegación principal, elija Etapas.
4. En la sección Registros y rastreo, elija Editar.
5. Active Registros de acceso personalizado.
6. En ARN del destino de registro de acceso, ingrese el ARN de un grupo de registro. El formato del ARN es `arn:aws:logs:{region}:{account-id}:log-group:log-group-name`.
7. En Formato de registro, ingrese un formato de registro. Puede elegir CLF, JSON, XML o CSV. Para obtener más información sobre ejemplos de formatos de registro, consulte the section called “Formatos de los registro de acceso”.

8. Seleccione Save changes (Guardar cambios).

API Gateway ya está listo para registrar solicitudes en la API. Si actualiza la configuración de la etapa, los registros o las variables de la etapa, no necesita volver a implementar la API.

Configuración del registro de acceso mediante CloudFormation

Utilice la siguiente plantilla de CloudFormation de ejemplo para crear un grupo de registro de Registros de Amazon CloudWatch y configurar el registro de acceso para una etapa. Para habilitar CloudWatch Logs, debe conceder permiso a API Gateway para leer y escribir registros en CloudWatch para su cuenta. Para obtener más información, consulte Asociar una cuenta a un rol de IAM en la Guía del usuario de AWS CloudFormation.

```
 TestStage:
   Type: AWS::ApiGateway::Stage
   Properties:
     StageName: test
     RestApiId: !Ref MyAPI
     DeploymentId: !Ref Deployment
     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
```

Registro de llamadas a la API en Amazon Data Firehose en API Gateway

Para ayudar a depurar problemas relacionados con el acceso de los clientes a la API, puede registrar llamadas a la API en Amazon Data Firehose. Para obtener más información acerca de Firehose, consulte What Is Amazon Data Firehose?.

Para el registro de acceso, solo puede habilitar CloudWatch o Firehose; no puede habilitar ambos. Sin embargo, puede habilitar CloudWatch para el registro de ejecución y Firehose para el registro de acceso.

Temas

- Formatos de registro de Firehose para API Gateway
- Permisos para el registro de Firehose
- Configuración del registro de acceso de Firehose mediante la consola de API Gateway

Formatos de registro de Firehose para API Gateway

El registro de Firehose utiliza el mismo formato que el registro de CloudWatch.

Permisos para el registro de Firehose

Si el registro de acceso de Firehose está habilitado en una etapa, API Gateway crea un rol vinculado al servicio en la cuenta si el rol no existe aún. Este rol se llama `AWSServiceRoleForAPIGateway` y tiene la política administrada `APIGatewayServiceRolePolicy` asociada. Para obtener más información acerca de los roles vinculados a servicio, consulte Uso de roles vinculados a servicios.

> **Note**
>
> El nombre del flujo de Firehose debe ser `amazon-apigateway-{your-stream-name}`.

Configuración del registro de acceso de Firehose mediante la consola de API Gateway

Para configurar el registro de API, la API debe estar implementada en una etapa. También debe haber creado un flujo de Firehose.

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Realice una de las siguientes acciones:
   a. Elija una API existente y después una etapa.
   b. Cree una API e impleméntela como etapa.
3. En el panel de navegación principal, elija Etapas.
4. En la sección Registros y rastreo, elija Editar.
5. Para habilitar el registro de acceso a un flujo de Firehose:

a. Active el Registro de acceso personalizado.

b. Para ARN del destino de registro de acceso, ingrese el ARN de un flujo de Firehose. El formato del ARN es `arn:aws:firehose:{region}:{account-id}:deliverystream/amazon-apigateway-{your-stream-name}`.

> **Note**
>
> El nombre del flujo de Firehose debe ser `amazon-apigateway-{your-stream-name}`.

c. En Formato de registro, ingrese un formato de registro. Puede elegir CLF, JSON, XML o CSV. Para obtener más información sobre ejemplos de formatos de registro, consulte the section called “Formatos de los registro de acceso”.

6. Seleccione Save changes (Guardar cambios).

API Gateway ya está preparado para registrar solicitudes a la API en Firehose. Si actualiza la configuración de la etapa, los registros o las variables de la etapa, no necesita volver a implementar la API.

Variables para el registro de acceso para API Gateway

En los registros de acceso, tanto usted como el desarrollador de la API pretenden registrar quién ha obtenido acceso a la API y cómo el intermediario ha obtenido acceso a la API. Puede crear su propio grupo de registros o elegir un grupo de registros existente, que podría administrarse a través de API Gateway. Para especificar los detalles de acceso, puede utilizar las siguientes variables `$context` que distinguen entre mayúsculas y minúsculas.

Para obtener una lista de variables de referencia para las transformaciones de datos, consulte the section called “Variables para transformaciones de datos”.

Parámetro	Descripción
`$context.accountId`	El ID de cuenta de AWS del propietario de la API.
`$context.apiId`	El identificador que API Gateway asigna a su API.

Parámetro	Descripción
`$context.authorize.error`	El mensaje de error de autorización.
`$context.authorize.latency`	La latencia de autorización en ms.
`$context.authorize.status`	El código de estado devuelto por un intento de autorización.
`$context.authorizer.claims.`*`property`	Una propiedad de las notificaciones devueltas por el grupo de usuarios de Amazon Cognito una vez que se ha autenticado correctamente al intermediario del método. Para obtener más información, consulte the section called “Uso de grupos de usuarios de Amazon Cognito como autorizador para una API REST”. **Note** La llamada a `$context.authorizer.claims` devuelve null.
`$context.authorizer.error`	El mensaje de error devuelto por un autorizador.
`$context.authorizer.integrationLatency`	La latencia de integración del autorizador en ms.
`$context.authorizer.integrationStatus`	El código de estado que devuelve un autorizador de Lambda.
`$context.authorizer.latency`	La latencia del autorizador en ms.

Parámetro	Descripción
`$context.authorizer.principalId`	La identificación de usuario principal asociada con el token enviado por el cliente y devuelto por el autorizador de Lambda de API Gateway (que anteriormente se denominaba autorizador personalizado). Para obtener más información, consulte the section called “Uso de autorizadores Lambda”.

`$context.authorizer.` *`property`*	El valor en forma de cadena del par clave-valor especificado de la asignación `context` que devuelve la función de Lambda del autorizador de Lambda de API Gateway. Por ejemplo, si el autorizador devuelve la siguiente asignación de `context`:
	`"context" : {`
	`"key": "value",`
	`"numKey": 1,`
	`"boolKey": true`
	`}`
	La llamada a `$context.authorizer.key` devuelve la cadena `"value"`, la llamada a `$context.authorizer.numKey` devuelve la cadena `"1"` y la llamada a `$context.authorizer.boolKey` devuelve la cadena `"true"`.
	En el caso de la *`propiedad`*, el único carácter especial admitido es el carácter `(_)` de subrayado.
	Para obtener más información, consulte the section called “Uso de autorizadores Lambda”.

`$context.authorizer.requestId`	El ID de solicitud del punto de enlace de AWS.

Parámetro	Descripción
`$context.authorizer.status`	El código de estado devuelto por un autorizador.
`$context.authenticate.error`	El mensaje de error devuelto por un intento de autorización.
`$context.authenticate.latency`	La latencia de autenticación en ms.
`$context.authenticate.status`	El código de estado devuelto por un intento de autenticación.
`$context.awsEndpointRequestId`	El ID de solicitud del punto de conexión de AWS.
`$context.cipherSuite`	El cifrado, en formato IANA, que se negocia durante el establecimiento de comunicación de TLS entre el cliente y API Gateway.
`$context.customDomain.basePathMatched`	La ruta de un mapeo de la API con la que coincidió una solicitud entrante. Aplicable cuando un cliente utiliza un nombre de dominio personalizado para acceder a una API. Por ejemplo, si un cliente envía una solicitud a `https://api.example.com/v1/orders/1234` y la solicitud empareja el mapeo de la API con la ruta `v1/orders`, el valor es `v1/orders`. Para obtener más información, consulte the section called “Mapeos de la API”.
`$context.customDomain.routingRuleIdMatched`	La regla de enrutamiento con la que coincide una solicitud entrante. Aplicable cuando un cliente utiliza un nombre de dominio personalizado para acceder a una API. Para obtener más información, consulte the section called “Reglas de enrutamiento”.

Parámetro	Descripción
`$context.deploymentId`	El ID de la implementación de la API.
`$context.domainName`	El nombre de dominio completo que se utiliza para invocar la API. Este debe ser el mismo que el encabezado `Host` entrante.
`$context.domainPrefix`	La primera etiqueta del `$context.domainName` .
`$context.endpointType`	El tipo del punto de conexión de la API.
`$context.error.message`	Una cadena que contiene un mensaje de error de API Gateway. Esta variable solo se puede usar para una sustitución sencilla de variables en una plantilla de mapeo de cuerpo GatewayResponse, que no procesa el motor de Velocity Template Language, y en el registro de acceso. Para obtener más información, consulte the section called “Métricas” y the section called “Configuración de respuestas de gateway para personalizar respuestas de errores”.
`$context.error.messageString`	El valor entrecomillado de `$context.error.message` , es decir, `"$context.error.message"` .

Parámetro	Descripción
`$context.error.responseType`	Un tipo de GatewayResponse. Esta variable solo se puede usar para una sustitución sencilla de variables en una plantilla de mapeo de cuerpo GatewayResponse, que no procesa el motor de Velocity Template Language, y en el registro de acceso. Para obtener más información, consulte the section called “Métricas” y the section called “Configuración de respuestas de gateway para personalizar respuestas de errores”.
`$context.error.validationErrorString`	Una cadena que contiene un mensaje de error de validación detallado.
`$context.extendedRequestId`	El ID ampliado que API Gateway genera y asigna a la solicitud de API. El ID de solicitud ampliado contiene información útil para la depuración y la resolución de problemas.
`$context.httpMethod`	El método HTTP utilizado. Los valores válidos son: `DELETE`, `GET`, `HEAD`, `OPTIONS`, `PATCH`, `POST` y `PUT`.
`$context.identity.accountId`	El ID de cuenta de AWS asociado con la solicitud.
`$context.identity.apiKey`	Para los métodos de API que necesitan una clave de API, esta variable es la clave de API asociada a la solicitud del método. Para métodos que no requieren una clave de API, esta variable corresponde a valores null. Para obtener más información, consulte the section called “Planes de uso”.
`$context.identity.apiKeyId`	El ID de clave de API asociado a una solicitud de API que requiere una clave de API.

Parámetro	Descripción
`$context.identity.caller`	El identificador principal del intermediario que firmó la solicitud. Compatible con recursos que utilizan la autorización de IAM.
`$context.identity.cognitoAuthenticationProvider`	Una lista separada por comas de todos los proveedores de autenticación de Amazon Cognito utilizados por el intermediario que realiza la solicitud. Solo está disponible si la solicitud se firmó con las credenciales de Amazon Cognito. Por ejemplo, para una identidad de un grupo de usuarios de Amazon Cognito, `cognito-idp.`*`region`*`.amazonaws.com/`*`user_pool_id`*`,cognito-idp.`*`region`*`.amazonaws.com/`*`user_pool_id`*`:CognitoSignIn:`*`token subject claim`* Consulte Uso de las identidades federadas en la Guía para desarrolladores de Amazon Cognito para obtener información sobre los proveedores de autenticación de Amazon Cognito disponibles.
`$context.identity.cognitoAuthenticationType`	El tipo de autenticación de Amazon Cognito del intermediario que realiza la solicitud. Solo está disponible si la solicitud se firmó con las credenciales de Amazon Cognito. Los valores posibles incluyen `authenticated` para identidades autenticadas y `unauthenticated` para identidades no autenticadas.
`$context.identity.cognitoIdentityId`	El ID de identidad de Amazon Cognito del intermediario que realiza la solicitud. Solo está disponible si la solicitud se firmó con las credenciales de Amazon Cognito.

Parámetro	Descripción
`$context.identity.cognitoIdentityPoolId`	El ID del grupo de identidades de Amazon Cognito del intermediario que realiza la solicitud. Solo está disponible si la solicitud se firmó con las credenciales de Amazon Cognito.
`$context.identity.principalOrgId`	El ID de organización de AWS.
`$context.identity.sourceIp`	La dirección IP de origen de la conexión TCP inmediata que realiza la solicitud al punto de conexión de API Gateway.
`$context.identity.clientCert.clientCertPem`	El certificado de cliente codificado en PEM que el cliente presentó durante la autenticación TLS mutua. Presente cuando un cliente accede a una API mediante un nombre de dominio personalizado que tiene una TLS mutua habilitada.
`$context.identity.clientCert.subjectDN`	Nombre distintivo del asunto del certificado que presenta un cliente. Presente cuando un cliente accede a una API mediante un nombre de dominio personalizado que tiene una TLS mutua habilitada.
`$context.identity.clientCert.issuerDN`	Nombre distintivo del emisor del certificado que presenta un cliente. Presente cuando un cliente accede a una API mediante un nombre de dominio personalizado que tiene una TLS mutua habilitada.
`$context.identity.clientCert.serialNumber`	Número de serie del certificado. Presente cuando un cliente accede a una API mediante un nombre de dominio personalizado que tiene una TLS mutua habilitada.

Parámetro	Descripción
`$context.identity.clientCert.validity.notBefore`	Fecha antes de la cual el certificado no es válido. Presente cuando un cliente accede a una API mediante un nombre de dominio personalizado que tiene una TLS mutua habilitada.
`$context.identity.clientCert.validity.notAfter`	Fecha después de la cual el certificado no es válido. Presente cuando un cliente accede a una API mediante un nombre de dominio personalizado que tiene una TLS mutua habilitada.
`$context.identity.vpcId`	El ID de VPC de la VPC que realiza la solicitud al punto de conexión de API Gateway.
`$context.identity.vpceId`	El ID de punto de conexión de VPC del punto de conexión de VPC que realiza la solicitud al punto de conexión de API Gateway. Está presente solo cuando tiene una API privada.
`$context.identity.user`	El identificador principal del usuario que se autorizará a acceder al recurso. Compatible con recursos que utilizan la autorización de IAM.
`$context.identity.userAgent`	El encabezado `User-Agent` del intermediario de la API.
`$context.identity.userArn`	El Nombre de recurso de Amazon (ARN) del usuario identificado después de la autenticación. Para obtener más información, consulte https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html.
`$context.integration.error`	El mensaje de error devuelto por una integración.

Parámetro	Descripción
`$context.integration.integrationStatus`	Para la integración de proxy de Lambda, el código de estado que se devuelve desde AWS Lambda, y no desde el código de función de Lambda del backend.
`$context.integration.latency`	La latencia de integración en ms. Es igual que `$context.integrationLatency` .
`$context.integration.requestId`	El ID de solicitud del punto de conexión de AWS. Es igual que `$context.awsEndpointRequestId` .
`$context.integration.responseTransferMode`	El modo de transferencia de respuesta de la integración. Puede ser `BUFFERED` o `STREAMED`.
`$context.integration.status`	El código de estado devuelto por una integración. Para integraciones de proxy de Lambda, este es el código de estado que devuelve su código de la función de Lambda.
`$context.integration.timeToAllHeaders`	El tiempo entre que API Gateway establece la conexión de integración y se reciben del cliente todos los encabezados de respuesta de integración.
`$context.integration.timeToFirstContent`	El tiempo transcurrido entre el momento en que API Gateway establece la conexión de integración y el momento en que recibe los primeros bytes de contenido.
`$context.integrationLatency`	La latencia de integración en ms.

Parámetro	Descripción
`$context.integrationStatus`	Para la integración de proxy de Lambda, este parámetro representa el código de estado que se devuelve desde AWS Lambda, y no desde el código de función de Lambda del backend.
`$context.isCanaryRequest`	Devuelve `true` si la solicitud se dirigió al canario y `false` si no se dirigió al canario. Está presente solo cuando tiene un canario activado.
`$context.path`	Ruta de acceso de la solicitud. Por ejemplo, en el caso del URL de una solicitud que no es de proxy `https://{rest-api-id}.execute-api.{region}.amazonaws.com/{stage}/root/child` , el valor de `$context.path` es `/{stage}/root/child` .
`$context.protocol`	Protocolo de la solicitud; por ejemplo, `HTTP/1.1`. **Note** Las API de API Gateway pueden aceptar solicitudes HTTP/2, pero API Gateway envía solicitudes a las integraciones de backend mediante HTTP/1.1. Como resultado, el protocolo de solicitud se registra como HTTP/1.1 incluso si un cliente envía una solicitud que usa HTTP/2.

Parámetro	Descripción
`$context.requestId`	ID de la solicitud. Los clientes pueden anular este ID de solicitud. Utilice `$context.extendedRequestId` para un ID de solicitud único que genera API Gateway.
`$context.requestOverride.header. header_name`	La invalidación del encabezado de solicitud. Si este parámetro se define, contiene los encabezados que se utilizarán en lugar de los HTTP Headers (Encabezados HTTP) que se definen en el panel Integration Request (Solicitud de integración). Para obtener más información, consulte Anulación de los parámetros de solicitud y respuesta y de los códigos de estado de la API para las API de REST en API Gateway.
`$context.requestOverride.path. path_name`	La invalidación de la ruta de acceso de la solicitud. Si este parámetro se define, contiene la ruta de acceso de la solicitud que se utilizará en lugar de los URL Path Parameters (Parámetros de ruta URL) que se definen en el panel Integration Request (Solicitud de integración). Para obtener más información, consulte Anulación de los parámetros de solicitud y respuesta y de los códigos de estado de la API para las API de REST en API Gateway.

Parámetro	Descripción
`$context.requestOverride.querystring.` *querystring_name*	La invalidación de la cadena de consulta de solicitud. Si este parámetro se define, contiene las cadenas de consulta de solicitud que se utilizarán en lugar de los URL Query String Parameters (Parámetros de cadena de consulta URL) que se definen en el panel Integration Request (Solicitud de integración). Para obtener más información, consulte Anulación de los parámetros de solicitud y respuesta y de los códigos de estado de la API para las API de REST en API Gateway.
`$context.responseLatency`	La latencia de respuesta en ms.
`$context.responseLength`	La duración de la carga de respuesta en bytes.
`$context.responseOverride.header.` *header_name*	La invalidación del encabezado de respuesta. Si este parámetro se define, contiene el encabezado que se devolverá en lugar del Response header (Encabezado de respuesta) que se define como Default mapping (Asignación predeterminada) en el panel Integration Response (Respuesta de integración). Para obtener más información, consulte Anulación de los parámetros de solicitud y respuesta y de los códigos de estado de la API para las API de REST en API Gateway.

Parámetro	Descripción
`$context.responseOverride.status`	La invalidación del código de estado de respuesta. Si este parámetro se define, contiene el código de estado que se devolverá en lugar del Method response status (Estado de respuesta de método) que se define como Default mapping (Asignación predeterminada) en el panel Integration Response (Respuesta de integración). Para obtener más información, consulte Anulación de los parámetros de solicitud y respuesta y de los códigos de estado de la API para las API de REST en API Gateway.
`$context.requestTime`	Hora de la solicitud en formato CLF-(`dd/MMM/yyyy:HH:mm:ss +-hhmm`).
`$context.requestTimeEpoch`	Hora de la solicitud en formato Epoch en milisegundos.
`$context.resourceId`	El identificador que API Gateway asigna a su recurso.
`$context.resourcePath`	La ruta a su recurso. Por ejemplo, en el caso del URI de una solicitud que no es de proxy `https://{rest-api-id}.execute-api.{region}.amazonaws.com/{stage}/root/child`, el valor de `$context.resourcePath` es `/root/child`. Para obtener más información, consulte Tutorial: Creación de una API de REST con integración no de proxy HTTP.
`$context.stage`	La etapa de implementación de la solicitud de la API (por ejemplo, `Beta` o `Prod`).
`$context.status`	El estado de respuesta de método.

Parámetro	Descripción
`$context.tlsVersion`	La versión de TLS que se negocia durante el establecimiento de comunicación de TLS entre el cliente y API Gateway.
`$context.waf.error`	El mensaje de error devuelto por AWS WAF.
`$context.waf.latency`	La latencia de AWS WAF en ms.
`$context.waf.status`	El código de estado devuelto por AWS WAF.
`$context.xrayTraceId`	La ID de rastreo para el rastro de X-Ray. Para obtener más información, consulte the section called “Configuración de AWS X-Ray”.
`$context.wafResponseCode`	La respuesta recibida desde AWS WAF: `WAF_ALLOW` o `WAF_BLOCK` . No se establecerá si la etapa no está asociada a una ACL web. Para obtener más información, consulte the section called “AWS WAF”.
`$context.webaclArn`	El ARN completo de la ACL web que se utiliza para decidir si se debe permitir o bloquear la solicitud. No se establecerá si la etapa no está asociada a una ACL web. Para obtener más información, consulte the section called “AWS WAF”.

Rastreo de las solicitudes de usuario a las API de REST mediante X-Ray en API Gateway

Puede utilizar AWS X-Ray para rastrear y analizar las solicitudes de los usuarios a medida que se trasladan a través de las API REST de Amazon API Gateway a los servicios subyacentes. API Gateway admite el rastreo de X-Ray para todos los tipos de punto de enlace de API de REST de API Gateway: regional, optimizado para bordes y privado. Puede utilizar X-Ray con Amazon API Gateway en todas las regiones de AWS donde X-Ray esté disponible.

Dado que X-Ray le ofrece la vista integral de una solicitud completa, puede analizar las latencias de sus API y sus servicios de backend. Puede utilizar un mapa de servicio de X-Ray para ver la latencia de una solicitud completa y la de los servicios posteriores que se integran con X-Ray. También puede configurar reglas de muestreo para indicar a X-Ray qué solicitudes debe registrar y a qué velocidad de muestreo, de acuerdo con los criterios que especifique.

Si llama a una API de API Gateway desde un servicio que ya se está rastreando, API Gateway transmite el rastreo, aunque dicho rastreo no esté habilitado en la API.

Puede habilitar X-Ray para una etapa de API mediante la consola de API Gateway o mediante la API o la CLI de API Gateway.

Temas

- Configuración de AWS X-Ray con las API de REST de API Gateway
- Uso de mapas de servicio y vistas de rastros de AWS X-Ray con API Gateway
- Configuración de reglas de muestreo de AWS X-Ray para las API de API Gateway
- Rastros de AWS X-Ray para las API de Amazon API Gateway

Configuración de AWS X-Ray con las API de REST de API Gateway

En esta sección, encontrará información detallada acerca de cómo configurar AWS X-Ray con las API REST de API Gateway.

Temas

- Modos de rastreo de X-Ray para API Gateway
- Permisos para el rastreo de X-Ray
- Activación del rastreo de X-Ray en la consola de API Gateway
- Habilitación del rastreo de AWS X-Ray mediante la CLI de API Gateway

Modos de rastreo de X-Ray para API Gateway

La ruta que recorre una solicitud a través de la aplicación se rastrea mediante un ID de rastro. Un rastro recopila todos los segmentos generados por una única solicitud, por lo general, una solicitud HTTP o `GET` `POST`.

Existen dos modos de rastreo para una API de API Gateway:

- Pasivo: esta es la configuración predeterminada si no ha habilitado el rastreo de X-Ray en una etapa de API. Este enfoque significa que la API de API Gateway solo se rastrea si X-Ray se ha activado en un servicio ascendente.
- Activo: cuando una etapa de API de API Gateway tiene esta configuración, API Gateway muestrea automáticamente las solicitudes de invocación de API, basándose en el algoritmo de muestreo especificado por X-Ray.

  Cuando el rastreo activo está habilitado en una etapa, API Gateway crea un rol vinculado al servicio en su cuenta, si el rol aún no existe. El rol se denomina `AWSServiceRoleForAPIGateway` y llevará asociada la política administrada `APIGatewayServiceRolePolicy`. Para obtener más información acerca de los roles vinculados a servicio, consulte Uso de roles vinculados a servicios.

  > **Note**
  >
  > X-Ray aplica un algoritmo de muestreo para garantizar que el rastreo sea eficiente, al tiempo que proporciona una muestra representativa de las solicitudes que recibe la API. El algoritmo de muestreo predeterminado es de una solicitud por segundo, con un muestreo del 5 por ciento de las solicitudes una vez pasado ese límite.

Puede cambiar el modo de rastreo de la API a través de la consola de administración de API Gateway, la CLI de API Gateway o un AWS SDK.

Permisos para el rastreo de X-Ray

Al habilitar el rastreo de X-Ray en una etapa, API Gateway crea un rol vinculado al servicio en su cuenta, si el rol aún no existe. El rol se denomina `AWSServiceRoleForAPIGateway` y llevará asociada la política administrada `APIGatewayServiceRolePolicy`. Para obtener más información acerca de los roles vinculados a servicio, consulte Uso de roles vinculados a servicios.

Activación del rastreo de X-Ray en la consola de API Gateway

Puede utilizar la consola de Amazon API Gateway para habilitar el rastreo activo en una etapa de API.

En estos pasos se presupone que ya ha implementado la API en una etapa.

1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.

2. Elija la API y, a continuación, en el panel de navegación principal, elija Etapas.
3. En el panel Etapas, elija una etapa.
4. En la sección Registros y rastreo, elija Editar.
5. Para habilitar el rastreo activo de X-Ray, seleccione Habilitar rastreo de X-Ray para activarlo.
6. Seleccione Save changes (Guardar cambios).

Una vez que haya habilitado X-Ray en la etapa de la API, puede utilizar la consola de administración de X-Ray para ver los rastros y los mapas de servicio.

Habilitación del rastreo de AWS X-Ray mediante la CLI de API Gateway

El siguiente comando create-stage permite crear una etapa con rastreo activo de X-Ray:

```
aws apigateway create-stage \
    --rest-api-id rest-api-id \
    --stage-name stage-name \
    --deployment-id deployment-id \
    --region region \
    --tracing-enabled=true
```

El resultado será similar al siguiente:

```
{
    "tracingEnabled": true,
    "stageName": stage-name,
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "deploymentId": deployment-id,
    "lastUpdatedDate": 1533849811,
    "createdDate": 1533849811,
    "methodSettings": {}
}
```

El siguiente comando create-stage permite crear una etapa sin rastreo activo de X-Ray:

```
aws apigateway create-stage \
    --rest-api-id rest-api-id \
    --stage-name stage-name \
```

```
    --deployment-id deployment-id \
    --region region \
    --tracing-enabled=false
```

El resultado será similar al siguiente:

```
{
    "tracingEnabled": false,
    "stageName": stage-name,
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "deploymentId": deployment-id,
    "lastUpdatedDate": 1533849811,
    "createdDate": 1533849811,
    "methodSettings": {}
}
```

El siguiente comando update-stage permite activar el rastreo activo de X-Ray para una API implementada:

```
aws apigateway update-stage \
    --rest-api-id rest-api-id \
    --stage-name stage-name \
    --patch-operations op=replace,path=/tracingEnabled,value=true
```

El siguiente comando update-stage permite desactivar el rastreo activo de X-Ray para una API implementada:

```
aws apigateway update-stage \
    --rest-api-id rest-api-id \
    --stage-name stage-name \
    --region region \
    --patch-operations op=replace,path=/tracingEnabled,value=false
```

El resultado será similar al siguiente:

```
{
    "tracingEnabled": false,
    "stageName": stage-name,
    "cacheClusterEnabled": false,
```

```
    "cacheClusterStatus": "NOT_AVAILABLE",
    "deploymentId": deployment-id,
    "lastUpdatedDate": 1533850033,
    "createdDate": 1533849811,
    "methodSettings": {}
}
```

Una vez que haya habilitado X-Ray para la etapa de su API, utilice la CLI de X-Ray para obtener información de rastreo. Para obtener más información, consulte Uso de la API de X-Ray con la AWS CLI.

Uso de mapas de servicio y vistas de rastros de AWS X-Ray con API Gateway

En esta sección, encontrará información detallada acerca de cómo utilizar los mapas de servicios y las vistas de rastros de AWS X-Ray con API Gateway.

Temas

- Ejemplo de mapa de servicio de X-Ray
- Ejemplo de vista de rastro de X-Ray

Ejemplo de mapa de servicio de X-Ray

AWS X-RayLos mapas de servicios de muestran información sobre la API y todos sus servicios posteriores. Cuando habilite X-Ray para una etapa de API en API Gateway, verá un nodo en el mapa de servicio que contiene información general sobre el tiempo total empleado en el servicio de API Gateway. Puede obtener información detallada sobre el estado de la respuesta y un histograma del tiempo de respuesta de la API para el periodo de tiempo seleccionado. En el caso de las API que se integran en servicios de AWS, como AWS Lambda y Amazon DynamoDB, verá más nodos con métricas de rendimiento relacionadas con esos servicios. Habrá un mapa de servicios para cada etapa de API.

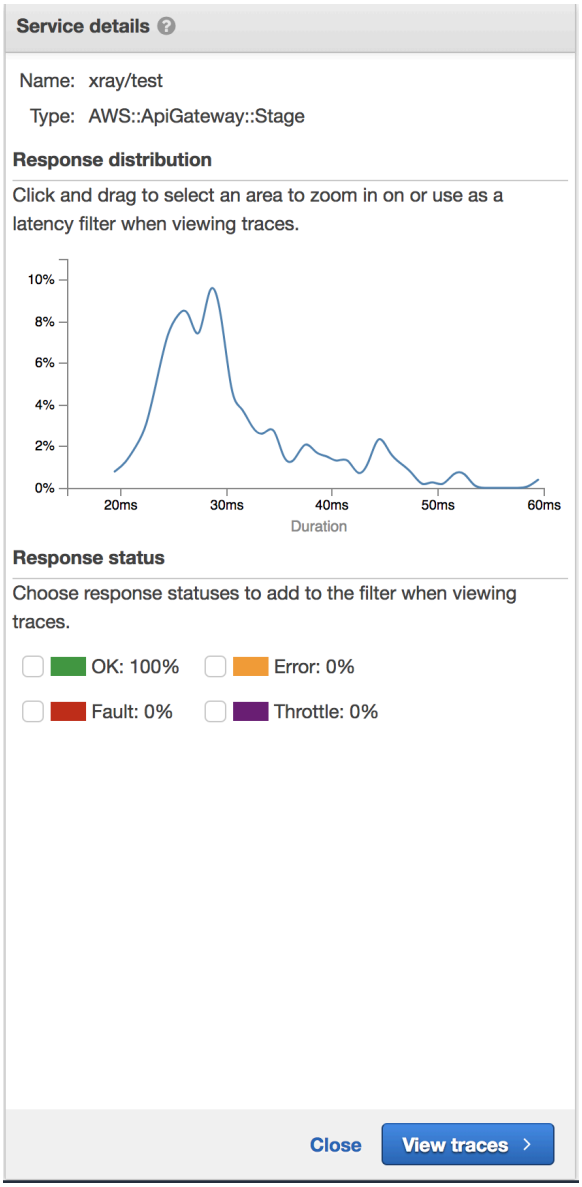
El siguiente ejemplo muestra un mapa de servicios para la etapa `test` de una API llamada `xray`. Esta API tiene dos integraciones de Lambda. Los nodos representan el servicio API Gateway y las dos funciones de Lambda.

Para obtener una explicación detallada de la estructura del mapa de servicio, consulte Uso del mapa de seguimiento de X-Ray.



Desde el mapa de servicios, puede ampliar el nivel de detalle para ver una vista de rastros de la etapa de la API. El rastro mostrará información detallada sobre la API, representada como segmentos y subsegmentos. Por ejemplo, el rastro del mapa de servicio mostrado arriba incluiría segmentos para el servicio de Lambda y la función de Lambda. Para obtener más información, consulte AWS Lambda y AWS X-Ray.

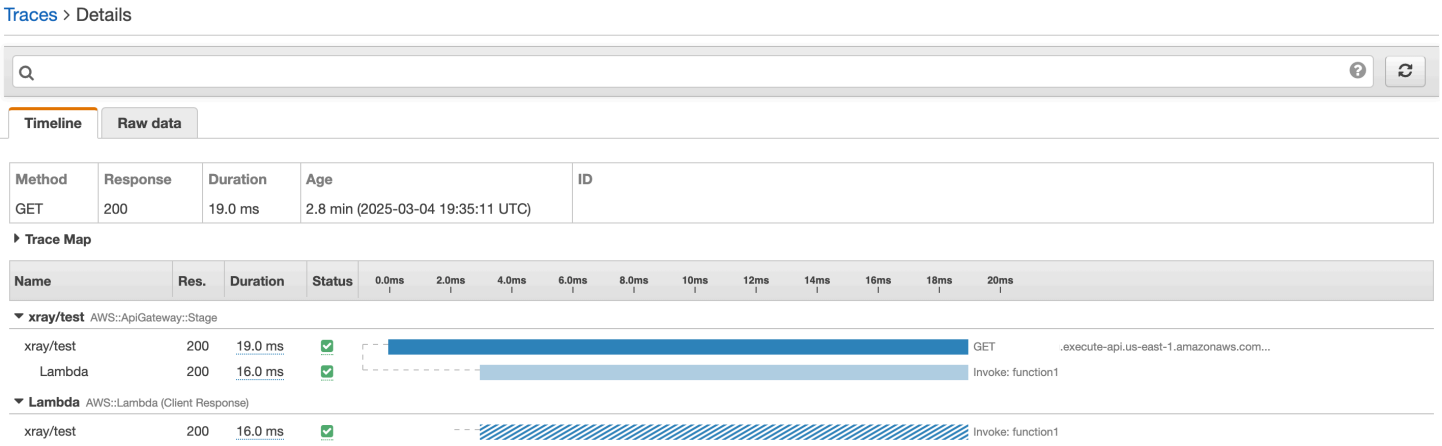
Si elige un nodo o borde en un mapa de servicio de X-Ray, la consola de X-Ray muestra un histograma de distribución de latencias. Puede utilizar un histograma de latencias para ver el tiempo que tarda un servicio en completar sus solicitudes. A continuación se presenta un histograma de la etapa de API Gateway denominada `xray/test` en el mapa de servicio anterior. Para obtener una explicación detallada de los histogramas de distribución de latencias, consulte Uso de los histogramas de latencias.



Ejemplo de vista de rastro de X-Ray

En el siguiente diagrama se muestra una vista de rastros generada para la API de ejemplo descrita anteriormente, con una función de backend de Lambda. Se muestra una solicitud de método de API realizada correctamente con un código de respuesta de 200.

Para obtener una explicación detallada de las vistas de rastros, consulte Visualización de rastros y detalles de rastros.



Configuración de reglas de muestreo de AWS X-Ray para las API de API Gateway

Puede utilizar la consola o el SDK de AWS X-Ray para configurar reglas de muestreo para su API de Amazon API Gateway. Una regla de muestreo especifica qué solicitudes debe registrar X-Ray para la API. Al personalizar las reglas de muestreo, puede controlar la cantidad de datos que va a registrar y modificar el comportamiento de muestreo sobre la marcha sin modificar o volver a implementar su código.

Antes de especificar sus reglas de muestreo de X-Ray, lea los siguientes temas en la guía para desarrolladores de X-Ray:

- Configuración de reglas de muestreo
- Uso de reglas de muestreo con la API de X-Ray

Temas

- Valores de opciones de regla de muestreo de X-Ray para las API de API Gateway
- Ejemplos de reglas de muestreo de X-Ray

Valores de opciones de regla de muestreo de X-Ray para las API de API Gateway

Las siguientes opciones de muestreo de X-Ray son relevantes para API Gateway. En los valores de cadena se pueden usar caracteres comodín para buscar coincidencias de un solo carácter (?) o cero o más caracteres (*). Para obtener más información, incluida una explicación detallada de la forma en que se usan las opciones Depósito y Porcentaje, consulte Configuración de reglas de muestreo.

- Rule name (Nombre de la regla) (cadena): un nombre único para la regla.

- Priority (Prioridad) (entero comprendido entre el 1 y 9999): prioridad de la regla de muestreo. Los servicios evalúan las reglas en orden ascendente de prioridad y toman una decisión de muestreo con la primera regla coincidente.
- Reservoir (Depósito) (entero no negativo): número fijo de solicitudes coincidentes que se van a instrumentar por segundo, antes de aplicar el porcentaje fijo. Los servicios no utilizan directamente el depósito, sino que se aplica a todos los servicios que usan la regla en su conjunto.
- Rate (Porcentaje) (número comprendido entre el 0 y el 100): porcentaje de solicitudes coincidentes que se van instrumentar, una vez que se ha agotado el depósito.
- Service name (Nombre de servicio) (cadena): nombre de la etapa de API, con el formato ***{api-name}*/*{stage-name}***. Por ejemplo, si implementara la API de ejemplo PetStore en una etapa llamada `test`, el valor de Service name (Nombre de servicio) que especificaría en la regla de muestreo sería **`pets/test`**.
- Service type (Tipo de servicio) (cadena): para una API de API Gateway, se puede especificar **`AWS::ApiGateway::Stage`** o **`AWS::ApiGateway::*`**.
- Host (Alojamiento) (cadena): nombre del alojamiento del encabezado del alojamiento de HTTP. Establezca esta opción en **`*`** para realizar la comparación con todos los nombres de host. También puede especificar un nombre de host completo o parcial para la comparación (por ejemplo, **`api.example.com`** o **`*.example.com`**).
- Resource ARN (ARN del recurso) (cadena): el ARN de la etapa de la API, por ejemplo, **`arn:aws:apigateway:`*`region`*`::/restapis/`*`api-id`*`/stages/`*`stage-name`***.

  El nombre de etapa se puede obtener de la consola, de la CLI o la API de API Gateway. Para obtener más información sobre los formatos de ARN, consulte la Referencia general de Amazon Web Services.
- HTTP method (Método HTTP) (cadena): método del que se va a realizar el muestreo (por ejemplo, **`GET`**).
- Ruta URL (cadena): la ruta URL de la solicitud.
- (opcional) Attributes (key and value) (Atributos: clave y valor): encabezados de la solicitud HTTP original (por ejemplo, **`Connection`**, **`Content-Length`** o **`Content-Type`**). Cada valor de atributo puede tener una longitud de hasta 32 caracteres.

Ejemplos de reglas de muestreo de X-Ray

Ejemplo de regla de muestreo #1

Esta regla realiza un muestreo de todas las solicitudes `GET` de la API `testxray` en la etapa `test`.

- Nombre de la regla — **`test-sampling`**
- Priority (Prioridad — **`17`**
- Reservoir size (Tamaño del depósito — **`10`**
- Fixed rate (Porcentaje fijo — **`10`**
- Service name (Nombre del servicio — **`testxray/test`**
- Service type (Tipo de servicio — **`AWS::ApiGateway::Stage`**
- HTTP method (Método HTTP — **`GET`**
- Resource ARN (ARN del recurso — **`*`**
- El host — **`*`**

Ejemplo de regla de muestreo #2

Esta regla realiza un muestreo de todas las solicitudes de la API `testxray` en la etapa `prod`.

- Nombre de la regla — **`prod-sampling`**
- Priority (Prioridad — **`478`**
- Reservoir size (Tamaño del depósito — **`1`**
- Fixed rate (Porcentaje fijo — **`60`**
- Service name (Nombre del servicio — **`testxray/prod`**
- Service type (Tipo de servicio — **`AWS::ApiGateway::Stage`**
- HTTP method (Método HTTP — **`*`**
- Resource ARN (ARN del recurso — **`*`**
- El host — **`*`**
- Attributes (Atributos — **`{}`**

Rastros de AWS X-Ray para las API de Amazon API Gateway

En esta sección, se analizan los segmentos, los subsegmentos y otros campos de los rastros de AWS X-Ray para las API de Amazon API Gateway.

Antes de leer esta sección, consulte los siguientes temas de la guía para desarrolladores de X-Ray:

- Uso de una Consola de administración de AWS
- Documentos de segmentos de X-Ray

- Conceptos

Temas

- Ejemplos de objetos de rastreo para una API de API Gateway
- Descripción del rastro

Ejemplos de objetos de rastreo para una API de API Gateway

En esta sección se explican algunos de los objetos que pueden aparecer en el rastro de una API de API Gateway.

Anotaciones

Las anotaciones pueden aparecer en segmentos y subsegmentos. Se utilizan como expresiones de filtro en reglas de muestreo para filtrar rastros. Para obtener más información, consulte Configuración de reglas de muestreo.

A continuación, se muestra un ejemplo de un objeto `annotations`, en el que una etapa de API se identifica mediante el ID de API y el nombre de etapa de la API:

```
"annotations": {
    "aws:api_id": "a1b2c3d4e5",
    "aws:api_stage": "dev"
}
```

Para obtener más información sobre las anotaciones, consulte Documentos del segmento de X-Ray y, a continuación, elija Documentos del segmento de X-Ray, Anotaciones.

AWS Datos de recursos de

El objeto `aws` solo aparece en los segmentos. A continuación se muestra un ejemplo de un objeto `aws` que coincide con la regla de muestreo personalizada. Para obtener una explicación detallada de las reglas de muestreo, consulte Configuración de reglas de muestreo.

```
"aws": {
    "xray": {
        "sampling_rule_name": "Default"
    },
    "api_gateway": {
        "account_id": "123412341234",
```

```
        "rest_api_id": "a1b2c3d4e5",
        "stage": "dev",
        "request_id": "a1b2c3d4-a1b2-a1b2-a1b2-a1b2c3d4e5f6"
    }
}
```

Para obtener más información sobre el objeto aws, consulte Documentos del segmento de X-Ray y, a continuación, elija Documentos del segmento de X-Ray, Datos de recursos de AWS.

Descripción del rastro

Lo que sigue es un segmento de rastreo para una etapa de API Gateway. Para obtener una explicación detallada de los campos que componen el segmento de rastro, consulte Documentos del segmento de X-Ray.

```
        {
            "Document": {
                "id": "a1b2c3d4a1b2c3d4",
                "name": "testxray/dev",
                "start_time": 1533928226.229,
                "end_time": 1533928226.614,
                "metadata": {
                    "default": {
                        "extended_request_id": "abcde12345abcde=",
                        "request_id": "a1b2c3d4-a1b2-a1b2-a1b2-a1b2c3d4e5f6"
                    }
                },
                "http": {
                    "request": {
                        "url": "https://example.com/dev?
username=demo&message=hellofromdemo/",
                        "method": "GET",
                        "client_ip": "192.0.2.0",
                        "x_forwarded_for": true
                    },
                    "response": {
                        "status": 200,
                        "content_length": 0
                    }
                },
                "aws": {
                    "xray": {
                        "sampling_rule_name": "Default"
```

```
                    },
                    "api_gateway": {
                        "account_id": "123412341234",
                        "rest_api_id": "a1b2c3d4e5",
                        "stage": "dev",
                        "request_id": "a1b2c3d4-a1b2-a1b2-a1b2-a1b2c3d4e5f6"
                    }
                },
                "annotations": {
                    "aws:api_id": "a1b2c3d4e5",
                    "aws:api_stage": "dev"
                },
                "trace_id": "1-a1b2c3d4-a1b2c3d4a1b2c3d4a1b2c3d4",
                "origin": "AWS::ApiGateway::Stage",
                "resource_arn": "arn:aws:apigateway:us-east-1::/restapis/a1b2c3d4e5/
stages/dev",
                "subsegments": [
                    {
                        "id": "abcdefgh12345678",
                        "name": "Lambda",
                        "start_time": 1533928226.233,
                        "end_time": 1533928226.6130002,
                        "http": {
                            "request": {
                                "url": "https://example.com/2015-03-31/functions/
arn:aws:lambda:us-east-1:123412341234:function:xray123/invocations",
                                "method": "GET"
                            },
                            "response": {
                                "status": 200,
                                "content_length": 62
                            }
                        },
                        "aws": {
                            "function_name": "xray123",
                            "region": "us-east-1",
                            "operation": "Invoke",
                            "resource_names": [
                                "xray123"
                            ]
                        },
                        "namespace": "aws"
                    }
                ]
```

```
            },
            "Id": "a1b2c3d4a1b2c3d4"
        }
```

Portales de API Gateway

Puede usar el portal de una API Gateway para compartir las API y la documentación de API con los consumidores de API en una página web. Un portal contiene un conjunto de productos de portal. Cada producto del portal es una agrupación lógica de API de REST y contiene la documentación que usted crea y publica para los consumidores de API. Las páginas de productos de un portal contienen la documentación personalizada por producto del portal. Las páginas de puntos de conexión REST de los productos contienen la documentación de cada una de las API de REST y la ruta, el método y la fase de una API de REST. La combinación de páginas de productos y páginas de puntos de conexión REST de productos proporciona la documentación completa que permite a los consumidores de API obtener información sobre cómo empezar a utilizar las API de REST.

Use un portal para hacer lo siguiente:

- Utilice API Gateway durante todo el ciclo de vida de las API, desde la creación de las API hasta su documentación y distribución.
- Comparta los productos del portal entre cuentas de AWS.
- Personalice y comparta una ubicación central para los productos del portal y proporcione páginas de productos.

Consideraciones

Es posible que las siguientes consideraciones afecten al uso de los portales de API Gateway:

- Solo puede crear portales para compartir las API de REST.
- Los portales se crean por región de AWS. El portal solo puede contener las API de REST que estén en la misma región en la que crea el portal.
- Solo puede usar los Consola de administración de AWS, AWS CLI o AWS SDK para crear y compartir los portales.
- Solo puede controlar el acceso al portal con grupos de usuarios de Amazon Cognito. Los consumidores del portal pueden iniciar sesión directamente a través de un grupo de usuarios o pueden federarse a través de un proveedor de identidades (IdP) externo. Si utiliza un grupo de usuarios de Amazon Cognito, no puede establecer un secreto de cliente para el grupo de usuarios. Para obtener más información, consulte Grupos de usuarios de Amazon Cognito.

Productos de portal en API Gateway

Un producto del portal representa un servicio o una funcionalidad que desea compartir. El producto del portal es una recopilación de puntos de conexión REST de productos y páginas de productos. Los puntos de conexión REST de productos son los puntos de acceso al producto del portal y constan de la ruta y el método de una API de REST y la fase en la que se implementa. Las páginas de productos son documentación que proporciona para explicar cómo los consumidores de API pueden utilizar los puntos de conexión de los productos. Un producto del portal puede contener toda la etapa `Prod` como una recopilación de puntos de conexión REST del producto o solo el recurso `GET /pets` implementado en la etapa `Prod` como un punto de conexión REST de un producto único.

El producto del portal es personalizable. Puede agregar documentación personalizada, cambiar el nombre de los puntos de conexión REST del producto, reorganizar el orden de visualización, agregar nuevas secciones y compartir productos entre cuentas de AWS. Para que los cambios que realice en el producto del portal surtan efecto, debe volver a publicar todos los portales que utilicen ese producto del portal.

Ejemplo de producto del portal de adopción de mascotas

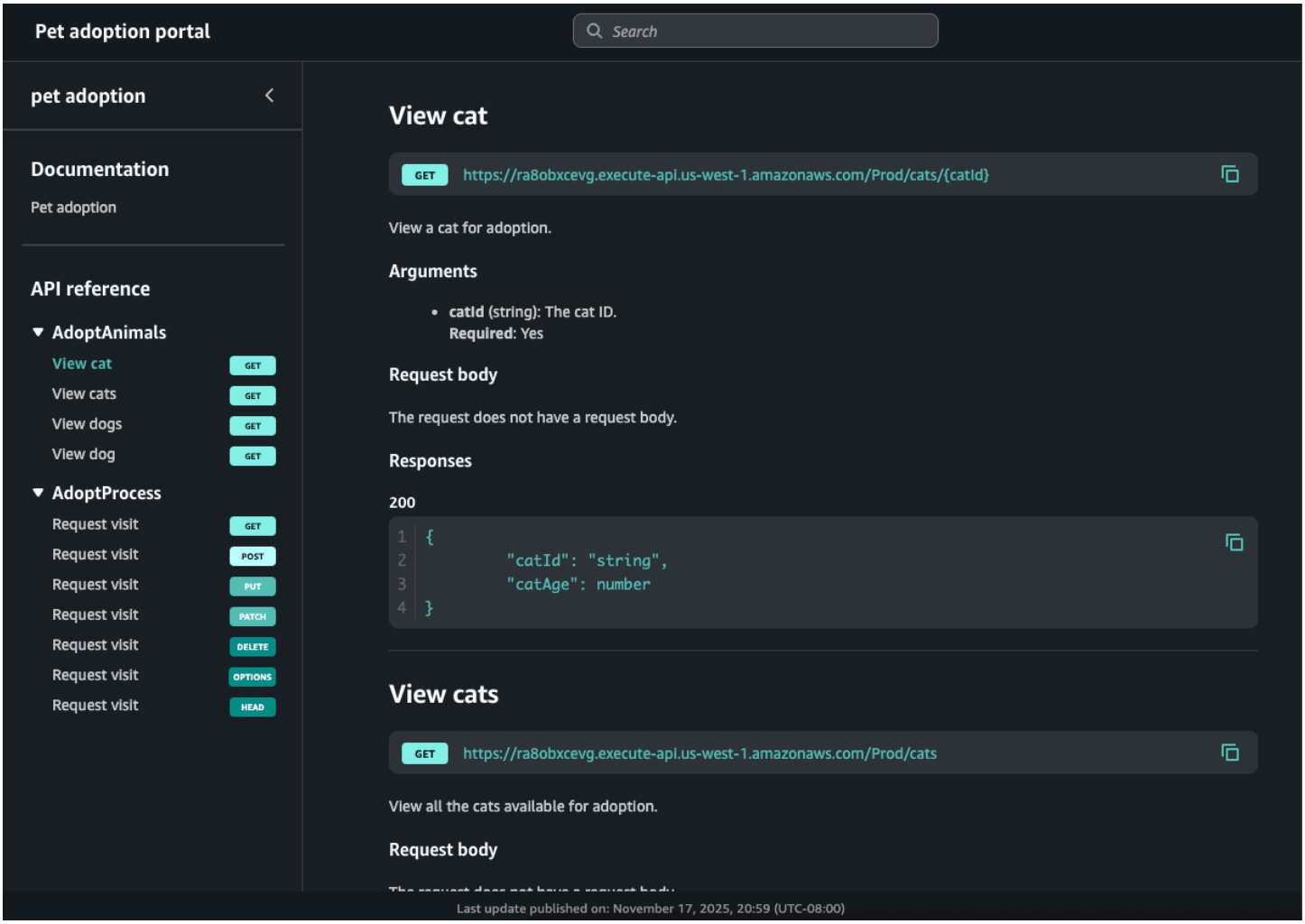
Por ejemplo, podría tener varias API de REST que representen un servicio de adopción de mascotas. Podría usar API Gateway para crear un producto del portal de `pet adoption`. Este producto del portal ayudaría a los clientes a detectar qué API deberían usar para conocer y adoptar mascotas. Este producto del portal utiliza las API de REST que ya ha creado, pero le permite reagruparlas y organizarlas. También puede proporcionar documentación sobre los términos y condiciones del uso del producto de portal de adopción de mascotas y permitir que los clientes prueben las API. Toda esta información se almacena en el producto del portal.

En la siguiente tabla se muestran tres API que representan un producto de portal de adopción de mascotas y los nombres de las operaciones de los puntos de conexión REST de los productos correspondientes y los nombres de las secciones de página.

ID de la API de REST	Ruta y método de la API de REST	Etapa de la API de REST	Nombre de operación	Sección de la página
`kf5387miad`	`GET /dogs`	Prod.	`View dogs`	`AdoptAnimals`

ID de la API de REST	Ruta y método de la API de REST	Etapa de la API de REST	Nombre de operación	Sección de la página
kf5387miad	GET /dogs/{dogId}	Prod.	View dog	AdoptAnimals
ra8obxcevg	GET /cats	Prod.	View cats	AdoptAnimals
ra8obxcevg	GET /cats/{catId}	Prod.	View cat	AdoptAnimals
h0rpx9cm62	ANY /user/{userId}/{petId+}	Beta	Request visit	AdoptProcess

En este ejemplo, dos API de REST, `ra8obxcevg` y `kf5387miad`, se agrupan en la sección `AdoptAnimals`. El resultado de ordenar la navegación tendría el siguiente aspecto en un portal:



Como el método ANY es un método general, el portal muestra todos los métodos HTTP compatibles. Este portal también contiene documentación creada por el propietario del producto del portal.

Siguientes pasos

Para empezar a utilizar los productos del portal, puede hacer lo siguiente:

- Para crear un producto del portal, consulte the section called “Creación de un producto del portal”.
- Para obtener información sobre la función de prueba, consulte the section called “Habilitación de una prueba para un punto de conexión REST de un producto en el portal”.
- Para obtener información sobre páginas del producto, consulte the section called “Creación de una página de producto”.
- Para obtener información sobre cómo compartir el producto del portal, consulte the section called “Uso compartido de productos de portal”.

Tras crear un producto de portal, puede publicarlo en un portal. Para obtener más información, consulte the section called “Creación de un portal”.

Creación de un producto del portal en API Gateway

El procedimiento siguiente muestra cómo crear un producto del portal. Un portal es una recopilación de productos del portal. Tras crear el producto del portal, cree puntos de conexión REST del producto y las páginas del producto. Para obtener información sobre los productos del portal, consulte the section called “Producto del portal”.

Consideraciones

Las siguientes consideraciones pueden afectar a cómo cree un producto del portal:

- El producto del portal puede contener API de REST públicas y privadas. Las API privadas no son compatibles con la funcionalidad de prueba y, por lo tanto, tienen una diferencia visual en el portal. Como propietario de un portal, es posible que tenga que proporcionar documentación para explicar esto.
- Si crea el producto del portal con la AWS CLI o los AWS SDK, el portal no tendrá puntos de conexión ni páginas de productos. Debe agregar estos recursos mediante la AWS CLI o la consola. Para obtener información sobre cómo crear un punto de conexión REST de producto, consulte the section called “Creación de un punto de conexión REST del producto”. Para obtener información sobre cómo crear una página de producto, consulte the section called “Creación de una página de producto”.

Creación de un producto del portal

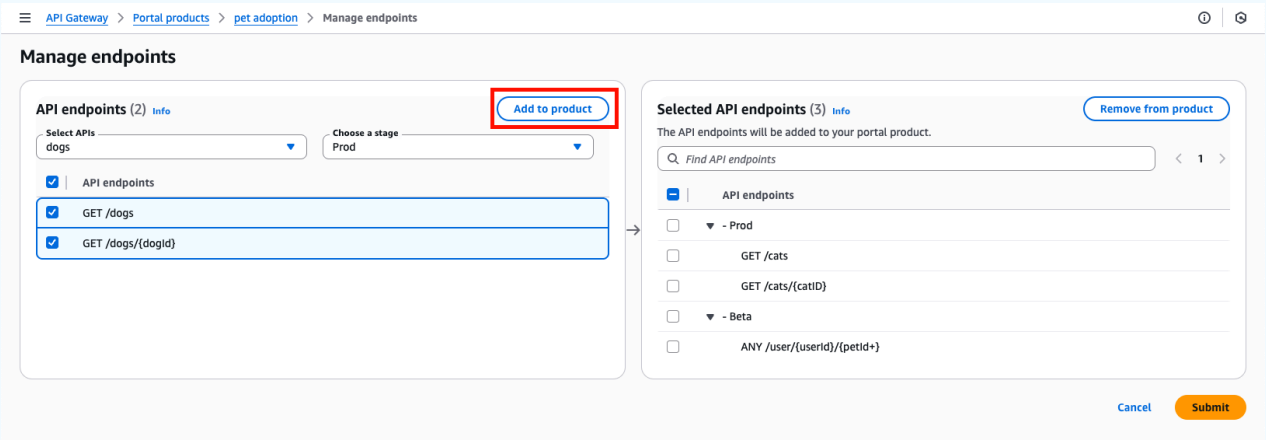
El procedimiento siguiente muestra cómo crear un producto del portal.

Creación de un producto del portal

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Productos de portales.
3. Seleccione Crear producto.
4. Para Nombre del producto, ingrese el nombre del producto del portal.
5. Para Descripción del producto, ingrese una descripción.
6. Elija Siguiente.

7. Para seleccionar los puntos de conexión REST del producto, en los puntos de conexión de la API, elija una API y, a continuación, elija una etapa.
8. Para agregar un punto de conexión a los puntos de conexión REST del producto, seleccione el punto de conexión de la API y, a continuación, elija Agregar al producto.

> **Note**
>
> No elija Siguiente sin antes elegir Agregar al producto.



   El punto de conexión de la API aparecerá en la lista de puntos de conexión de la API seleccionados.
9. Elija Siguiente.
10. Revise la selección y elija Crear producto.

Después de crear el producto del portal mediante la consola, todas las páginas del producto y las páginas de punto de conexión REST del producto son borradores y no aparecerán en el portal. Para que las páginas de los productos y las páginas de punto de conexión de REST de los productos estén visibles para los consumidores, debe agregar el borrador a una sección. Si crea el producto del portal con la AWS CLI o los AWS SDK, agregue el borrador a la sección en el comando AWS CLI. Independientemente de cómo agrega el borrador a una sección, debe publicar el portal que utiliza los productos del portal para que los consumidores puedan verlo.

Agregación del borrador a una sección de la página

1. Los borradores se muestran en la pestaña Documentación. Hay borradores de páginas de documentación para las páginas de productos y borradores de páginas de referencia de API

para las páginas de punto de conexión REST de los productos. Elija Borradores de páginas de referencia de la API.

2. Elija un borrador de páginas de referencia de la API.

   Si no tiene ningún punto de conexión REST de productos, no tendrá ningún borrador de las páginas de referencia de la API. Para obtener información sobre cómo crear un punto de conexión REST de producto, consulte the section called “Creación de un punto de conexión REST del producto”.
3. Elija Editar página.
4. En esta página, puede sobrescribir cualquier parte de la documentación de la API existente o utilizar la documentación de API Gateway. Para que los consumidores puedan ver el contenido de la página de punto de conexión REST del producto, ingrese un nombre en Nombre de sección. Si este fuera el punto de conexión de `/pets-GET`, el nombre de la página podría ser `Pets`.
5. Seleccione Save changes (Guardar cambios).
6. El nombre de la nueva página que ha creado aparece en la sección páginas de referencia de la API.

   Para que los consumidores puedan ver la nueva página, igual tendrá que volver a publicar el portal. Para obtener más información, consulte the section called “Publicación de un portal”.

Creación de un punto de conexión REST del producto en API Gateway

Un punto de conexión REST de un producto es un punto de acceso al producto del portal. Cada punto de conexión consta de la ruta y el método de una API de REST y de la fase en la que se implementa. El punto de conexión incluye toda la documentación que haya definido para la API de REST o se puede actualizar con documentación personalizada. Al crear el punto de conexión REST del producto, se configuran los siguientes ajustes:

- El nombre de la operación del punto de conexión REST del producto. Si la API de REST tiene la ruta y el método `GET /pets`, en un portal, el nombre es `/pets`. Con el nombre de la operación, puede cambiar el nombre del punto de conexión REST del producto a `Pet summaries`. Esto no afectará a la API existente.
- La función de prueba para el punto de conexión del producto. Esto permite a los consumidores de API probar las API en el portal. Si crea un punto de conexión de producto para una API privada, la función de prueba se establece en `UNAVAILABLE`. Para obtener más información, consulte the

section called “Habilitación de una prueba para un punto de conexión REST de un producto en el portal”.

- Si actualmente tienes partes de documentación definidas para la API de REST, puede importar la documentación o invalidar la documentación existente y crear nueva documentación por producto mediante GitHub Flavored Markdown. Las invalidaciones de documentación no se propagan a las partes de la documentación de la API de API Gateway. Si importa las partes de la documentación actual, API Gateway sincroniza los cambios en la API con el punto de conexión REST del producto.

Creación de un punto de conexión REST del producto

En el siguiente procedimiento, se muestra cómo agregar un punto de conexión REST de producto a un producto existente del portal. Para obtener información sobre cómo crear un producto de portal y un punto de conexión REST de producto juntos, consulte the section called “Creación de un producto del portal”.

Creación de un punto de conexión REST de un producto para un portal de productos

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Productos de portales.
3. Elija un producto del portal.
4. Elija la pestaña Puntos de conexión asociados y, a continuación, elija Agregar puntos de conexión.
5. Para seleccionar los puntos de conexión REST del producto, en los puntos de conexión de la API, elija una API y, a continuación, elija una etapa.
6. Para agregar un punto de conexión a los puntos de conexión REST del producto, seleccione el punto de conexión de la API y, a continuación, elija Agregar al producto.

   El punto de conexión de la API aparecerá en la lista de puntos de conexión de la API seleccionados.
7. Elija Enviar.
8. La página de punto de conexión REST del producto es ahora un borrador. Elija Borradores de páginas de referencia de la API y, a continuación, elija la página de punto de conexión REST del producto para finalizar el contenido de la página.
9. Elija Editar página.

10. Para invalidar la documentación existente de API Gateway, en Origen de documentación, active Invalidar la documentación y haga lo siguiente:

    a. Para Mostrar contenido, elija Crear invalidación.
    b. Para Nombre de operación, ingrese un nombre de operación nuevo.
    c. Para Cuerpo de página, ingrese la documentación personalizada.
    d. (Opcional) Para Punto de conexión, ingrese un punto de conexión personalizado que aparezca en el portal.
11. Para permitir que los clientes invoquen la API en el portal, seleccione la funcionalidad de la prueba. Para obtener más información, consulte the section called “Habilitación de una prueba para un punto de conexión REST de un producto en el portal”.
12. Para Nombre de sección, ingrese un nombre o elija una sección existente. Al agregar el punto de conexión REST del producto a una sección, los consumidores pueden ver el contenido en un portal publicado.
13. Seleccione Save changes (Guardar cambios).

Para que los consumidores puedan ver la nueva página, igual tendrá que volver a publicar el portal. Para obtener más información, consulte the section called “Publicación de un portal”.

Creación de una página de producto en API Gateway

Una página del producto ayuda a los consumidores de los productos a entender y utilizar los productos. La documentación se encuentra por producto, por lo que todos los portales que incluyan el producto disponen de esta documentación. API Gateway admite dos tipos de páginas de productos escritas en GitHub Flavored Markdown:

Página de información general

Al crear el producto, API Gateway genera automáticamente una página de Información general del producto en función de la información que proporciona al crear el producto. Esta página tiene un formato adicional que no se puede cambiar. Puede modificar el texto o eliminar la página del producto del portal.

Páginas personalizadas

Una página personalizada es una página escrita en Markdown. Puede usar esta sección para ver los términos y condiciones o un tutorial sobre cómo usar el producto. Puede convertir cualquier página personalizada en una página de información general. Al convertir una página

personalizada en una página de información general, el contenido de la página tiene un formato que coincide con el de la página de información general.

Consideraciones

Las siguientes consideraciones pueden afectar al uso de las páginas de productos:

- Esta documentación no se encuentra por API y no incluye ningún método, recurso ni parámetro de solicitud. Para obtener información sobre cómo modificar la documentación por API de un producto, consulte the section called “Actualización de un punto de conexión REST del producto”.
- No puede cargar un archivo `.md` directamente en el producto.
- Puede modificar el orden de las páginas de productos modificando la lista de páginas en el orden de visualización del producto del portal. Para obtener más información, consulte the section called “Actualización de un producto del portal”.
- Debe volver a publicar todos los portales que incluyan el producto para que los cambios en la documentación surtan efecto.
- Si tiene varios portales que utilizan el mismo producto del portal, las páginas del producto son las mismas en todos los portales. Para tener los mismos puntos de conexión REST del producto pero diferentes páginas de productos, cree varios productos.

Creación de una página de producto

El siguiente procedimiento muestra cómo crear una página de producto.

Creación de una página de producto

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Productos de portales.
3. Elija un producto.
4. En Documentación, elija Crear página personalizada.
5. Para Título de la página, ingrese el título de la página.
6. Para Cuerpo, ingrese la documentación complementaria mediante GitHub Flavored Markdown.

   La sección de vista previa muestra cómo aparece el contenido en un portal. El estilo visual final puede cambiar en función de la configuración del portal.
7. Elija Crear página del producto.

8. La página del producto es ahora un borrador. Elija Borradores de páginas de documentación y, a continuación, elija la página del producto para finalizar el contenido de la página.
9. Elija Editar página.
10. Para Nombre de sección, ingrese un nombre o elija una sección existente. Al agregar la página del producto a una sección, los consumidores pueden ver el contenido en un portal publicado.
11. Seleccione Save changes (Guardar cambios).

Para que los consumidores puedan ver la nueva página, igual tendrá que volver a publicar el portal. Para obtener más información, consulte the section called “Publicación de un portal”.

Actualización de un producto del portal en API Gateway

Al actualizar un producto del portal, puede cambiar el nombre, la descripción o el orden de visualización de cómo aparecen los puntos de conexión REST del producto y las páginas de productos. Para modificar el orden de visualización, modifique la lista de páginas para las páginas de productos y el nombre de la sección para los puntos de conexión de los productos. Si lo hace mediante la AWS CLI o los SDK, especifique el ARN de la página del producto o el ARN del punto de conexión REST del producto. Debe volver a publicar todos los portales que utilicen los productos para que los consumidores de la API puedan ver los cambios reflejados en un portal.

Actualización de un producto del portal

En el siguiente procedimiento se muestra cómo cambiar la lista de secciones para reordenar los puntos de conexión REST del producto tal como aparecerán en un portal. En este procedimiento, asumimos que tiene al menos dos puntos de conexión REST de producto en el producto del portal.

Actualización de un producto del portal

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Productos de portales.
3. Elija un producto.
4. En la pestaña Documentación, en las páginas de productos, elija Administrar.
5. Arrastre y suelte los puntos de conexión REST del producto para reordenarlos. También puede mover los puntos de conexión REST del producto y las páginas del producto a la sección de borradores de páginas. Las páginas del borrador de las páginas no estarán visibles en el portal.
6. (Opcional) Cambie el nombre de los puntos de conexión REST del producto o agregue una nueva sección de referencia de la API. Estos cambios no afectarán a las API de REST.

7. Seleccione Save changes (Guardar cambios).

Actualización de un punto de conexión REST del producto en API Gateway

Al actualizar un punto de conexión REST de un producto, puede cambiar todos los ajustes del punto de conexión del producto, excepto la ruta y el método de una API de REST y la fase en la que se implementa. Todos los cambios en los puntos de conexión de los productos se actualizan automáticamente, pero debe volver a publicar todos los portales que utilicen los productos para ver los cambios reflejados en un portal.

Actualización de un punto de conexión REST del producto

El siguiente procedimiento muestra cómo actualizar un punto de conexión REST del producto para sobrescribir la documentación de la API actual con una documentación de referencia de la API personalizada.

Actualización de un punto de conexión REST del producto

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Productos de portales.
3. Elija un producto.
4. En la pestaña Documentación, en las páginas de referencia de la API, elija el nombre del punto de conexión REST de un producto, como /dogs - GET.
5. Para Vista previa, elija Editar página.
6. Para Origen de la documentación, active Invalidación de la documentación.
7. En Mostrar contenido, elija Invalidación del contenido existente.

   Si elige Eliminar todo el contenido, el contenido se elimina de la página, pero puede elegir Invalidar el contenido existente para volver a acceder a la documentación.
8. En el cuerpo de la página, ingrese la documentación de referencia de la API personalizada mediante GitHub Flavored Markdown.

   Esta documentación no se volverá a propagar a API Gateway.
9. Seleccione Save changes (Guardar cambios).
10. Para que los cambios surtan efecto, es necesario volver a publicar cualquier portal que utilice este producto de portal.

Actualización de una página de producto en API Gateway

Cuando actualice la página de un producto, puede cambiar todos los ajustes de la página del producto. Todos los cambios en la página del producto se actualizan automáticamente, pero debe volver a publicar todos los portales que utilicen los productos para ver los cambios reflejados en un portal.

Uso compartido de productos de portal en API Gateway

Puede compartir un producto del portal a través de Cuentas de AWS mediante AWS RAM. Cuando comparte un producto del portal, la otra cuenta puede usar el producto del portal en su propio portal. Con los productos del portal compartidos, puede crear un catálogo único de las API de la organización y hacer cumplir los estándares de gobernanza en todo su ecosistema de API. Al mismo tiempo, compartir los productos del portal proporciona flexibilidad a los proveedores de la API para desarrollar, probar y mantener las API en sus propias cuentas.

Consideraciones

Las siguientes consideraciones pueden afectar a cómo comparte los recursos del portal:

- Cuando comparte el producto con otra cuenta, esa cuenta no puede modificar ninguna propiedad de la API de REST. Esto incluye los puntos de conexión de la integración, la estrategia de autorización o la configuración de las etapas.
- Cuando agrega el producto del portal de otra cuenta al portal, el propietario del producto del portal no puede ver ni controlar ninguna otra propiedad del portal. El propietario del producto del portal solo sabe que el producto se está utilizando en el portal.
- Los productos del portal de API Gateway se comparten por Región de AWS.
- Puede usar un recurso compartido con varias entidades principales y, una vez creado el recurso compartido, puede agregarle más entidades principales. Le recomendamos que reutilice el recurso compartido siempre que sea posible.
- Si ambas cuentas pertenecen a la misma organización que utiliza AWS Organizations, el recurso compartido se acepta automáticamente. Aún en este caso debe crear el recurso compartido mediante AWS RAM.
- Si ambas cuentas pertenecen a la misma organización que utiliza AWS Organizations y la organización tiene habilitado el uso compartido de recursos, se concederá automáticamente acceso a los recursos compartidos a todas las entidades principales de la organización con las que comparte. No se precisa invitación y puede omitir el uso compartido de recursos.

- Si la cuenta con la que compartió el producto no acepta el recurso compartido en un plazo de 12 horas, debe volver a compartir el recurso.
- Tras crear el recurso compartido, AWS RAM actualiza la política de uso compartido de productos del producto para impedir el acceso a las entidades principales que no dispongan de un acceso `allow` explícito. Para obtener más información, consulte Cómo determinar si una solicitud se permite o se deniega dentro de una cuenta en la Guía del usuario de IAM.

  La política de recursos actualizada será similar a lo siguiente:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "apigateway:GetProduct",
                "apigateway:ListEndpoints",
                "apigateway:ListPages",
                "apigateway:GetEndpoint",
                "apigateway:GetPage"
            ],
            "Resource": [
                "arn:aws:apigateway:us-east-1:111122223333:/portalproducts/product-id",
                "arn:aws:apigateway:us-east-1:111122223333:/portalproducts/product-id/*"
            ]
        }
    ]
}
```

  AWS RAM ha impedido que las entidades principales sin acceso allow explícito agreguen su producto a su portal al agregar lo siguiente:

```
"StringNotEquals": { "aws:PrincipalAccount": "555555555555" }
```

Para obtener información sobre cómo compartir un producto, consulte the section called “Compartir el producto de portal con el propietario del portal”. Para obtener información sobre cómo agregar al

portal un producto compartido con usted, consulte the section called “Agregación de un producto del portal compartido al portal”.

Compartir el producto del portal con el propietario del portal en API Gateway

Como propietario de un producto, puede compartir el producto con el propietario del portal en otra Cuenta de AWS mediante AWS RAM. Se mantiene el control total de todas las páginas de productos y de las páginas de punto de conexión REST de los productos.

Consideraciones

Las siguientes consideraciones pueden afectar a cómo comparte los productos del portal:

- Puede compartir o dejar de compartir los productos del portal en cualquier momento. Si deja de compartir un producto del portal mientras se está utilizando en un portal publicado, el propietario del portal no podrá ver el producto del portal, ver las actualizaciones que se le hayan realizado ni volver a publicar ni obtener una vista previa del portal hasta que elimine el producto del portal.
- Puede ver los productos del portal que ha compartido con otras cuentas.
- No puede modificar ningún portal que contenga el producto del portal, a menos que cree el portal usted mismo.

Uso compartido del producto del portal

El procedimiento siguiente muestra cómo crear un recurso compartido.

Consola de administración de AWS

Para usar la Consola de administración de AWS, consulte Creación de un recurso compartido en AWS RAM en la Guía del usuario de AWS RAM.

En Seleccionar tipo de recurso, elija Producto de API Gateway.

AWS CLI

El siguiente comando create-resource-share permite crear un recurso compartido para el nombre de dominio personalizado privado. La asociación del recurso y la entidad principal puede tardar unos minutos en completarse. Para las entidades principales, proporcione un ID de cuenta o un ID de Organizations, como `arn:aws:organizations::123456789012:organization/o-1234abcd`. Puede proporcionar varias entidades principales para el recurso compartido.

```
aws ram create-resource-share \
 --region us-west-2 \
 --name portal-product-resource-share \
 --permission-arns arn:aws:ram::aws:permission/
AWSRAMDefaultPermissionAPIGatewayDeveloperPortalProduct \
 --resource-arns arn:aws:apigateway:us-west-2:111122223333:/portalproducts/
p000000000 \
 --principals 222222222222
```

Para dejar de compartir el producto del portal, utilice AWS RAM para eliminar el recurso compartido.

En cualquier momento, puede modificar la política de uso compartido de productos para modificar qué entidades principales pueden utilizar los productos del portal en sus portales.

Actualización de la política de uso compartido de productos

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Productos de portales.
3. Elija un producto.
4. En la pestaña Uso compartido de productos, en Política de uso compartido de productos, elija Administración de política de uso compartido de productos.
5. Si no ha compartido el producto anteriormente, la política de uso compartido de productos denegará el acceso de otras cuentas a los productos del portal. Debe actualizar la política para permitir el acceso de determinadas cuentas al producto del portal.
6. Tras actualizar la política de uso compartido de productos, elija Guardar cambios.

Agregación de un producto del portal compartido al portal en API Gateway

Como propietario del portal, puede usar productos del portal compartidos con usted por otras cuentas en el portal. Mantenga pleno control del portal.

Consideraciones

Las siguientes consideraciones pueden afectar a cómo comparte los recursos del portal:

- Debe aceptar el recurso compartido del producto del propietario del producto para poder utilizar el producto en el portal. Si el propietario del producto y usted están en la misma organización, AWS RAM puede completar algunos pasos compartidos para usted.

- Si el propietario del producto deja de compartir un producto mientras está en el portal publicado, el producto seguirá visible en el portal, pero no podrá volver a publicar el portal hasta que lo elimine. Como propietario del portal, es su responsabilidad eliminar el producto del portal, ya que el propietario del producto no puede hacerlo por usted.
- Puede ver los productos del portal, las páginas de productos y las páginas de punto de conexión REST del producto, pero no puede modificar ninguno de estos recursos.
- Si se comparte un producto con usted, no podrá compartirlo con otra cuenta.

(Opcional) Aceptación del recurso compartido

Después de que el propietario del producto cree un recurso compartido, dispone de 12 horas para aceptarlo. Si pertenece a la misma organización que utiliza AWS Organizations como el propietario del producto, el recurso compartido se acepta automáticamente. Si pertenece a una organización que tiene habilitados los recursos compartidos automáticos, el recurso se comparte automáticamente con usted.

Consola de administración de AWS

Para usar la Consola de administración de AWS, consulte Aceptar y rechazar invitaciones a recursos compartidos en la Guía del usuario de AWS RAM.

AWS CLI

Para buscar todos los recursos que se comparten con usted, utilice el siguiente comando get-resource-share-invitations:

```
aws ram get-resource-share-invitations \
    --region us-west-2
```

Utilice el ARN del recurso compartido resultante para aceptar la invitación al recurso compartido. El siguiente comando accept-resource-share-invitation permite aceptar el recurso compartido.

```
aws ram accept-resource-share-invitation \
    --resource-share-invitation-arn arn:aws:ram:us-west-2:123456789012:resource-
share-invitation/1e3477be-4a95-46b4-bbe0-c4001EXAMPLE \
    --region us-west-2
```

Agregación de un producto compartido en el portal

Tras aceptar el recurso compartido, agregue el producto al portal.

Consola de administración de AWS

Agregación de un producto compartido al portal

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Portales.
3. Elija un portal.
4. En la pestaña Productos, para los productos del portal, elija Agregar productos.
5. Agregue un producto al portal. Los productos compartidos se muestran como `shared` en la lista de productos.
6. Seleccione Save changes (Guardar cambios).

AWS CLI

Para encontrar todos los productos del portal que se han compartido con usted, utilice el comando `get-portal-products` siguiente:

```
aws apigatewayv2 get-portal-products \
    --resource-owner OTHER_ACCOUNTS \
    --region us-west-2
```

Para agregar un producto de portal compartido al portal, utilice el comando `update-portal` siguiente:

```
aws apigateway update-portal \
    --included-portal-product-arns arn:aws:apigateway:us-west-2:111122223333:/
portalproducts/p000000000 \
    --region us-west-2
```

Habilitación de una prueba para un punto de conexión REST de un producto de API Gateway en el portal

Use la prueba para permitir que un consumidor de API invoque el punto de conexión del producto desde el portal. Cuando un consumidor de la API utiliza la prueba, ingresa los parámetros de solicitud del método e invoca el punto de conexión del producto. A continuación, API Gateway invoca la API a través de Internet pública y devuelve la respuesta en la ventana de la prueba. Puede usar una página de producto personalizada para proporcionar cualquier información adicional sobre los parámetros necesarios para acceder a la API.

API Gateway utiliza los siguientes límites para proteger las API:

- API Gateway solo permite 3 solicitudes por segundo a la API.
- API Gateway utiliza un límite de tiempo de espera integrado de 29 000 ms. Es posible que la API actual tenga un límite de tiempo de espera más alto, pero API Gateway no aplica este tiempo de espera cuando un cliente usa la prueba.
- API Gateway limita la carga útil de respuesta a 6 MB.

Consideraciones

Las siguientes consideraciones pueden afectar a cómo usar la prueba:

- La prueba no es compatible con la vista previa de un portal.
- La prueba no es compatible para las API de REST con las siguientes características:
  - API privadas
  - API que utilizan TLS mutuo
  - API que utilizan certificados SSL privados o autofirmados

  Como propietario del portal, es responsable de la comunicación con los consumidores de la API. El motivo por el que usar el botón de prueba no existe para ninguna API de REST que no sea compatible. API Gateway no le explica esto.

Habilitación de la prueba para un punto de conexión REST de un producto

En el siguiente procedimiento, se muestra cómo habilitar la prueba para un punto de conexión de producto.

Para obtener información sobre cómo utilizar la prueba en un portal, consulte the section called “Uso de un portal”.

Habilitación de la prueba para un punto de conexión REST de un producto

1. En el panel de navegación principal, elija Productos de portales.
2. Elija un producto.
3. En la pestaña Documentación, en las páginas de referencia de la API, elija el nombre del punto de conexión REST de un producto, como /dogs - GET.
4. Elija Editar página.
5. Para permitir que los clientes invoquen la API en el portal, seleccione la funcionalidad de la prueba.
6. Seleccione Save changes (Guardar cambios).
7. Para que los cambios surtan efecto, es necesario volver a publicar cualquier portal que utilice este producto de portal.

Eliminación de un producto del portal en API Gateway

Al eliminar un producto del portal, no pude recuperarse.

Eliminación de un producto del portal

El procedimiento siguiente muestra cómo eliminar un producto del portal.

Eliminación de un producto del portal

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Productos de portales.
3. Elija un producto del portal.
4. Elija Eliminar.
5. Confirme la elección y elija Eliminar.
6. Para que los cambios surtan efecto, es necesario volver a publicar cualquier portal que utilice este producto de portal.

Creación de un portal en API Gateway

Un portal es una recopilación de productos. Usted controla la autorización, la marca y la publicación del portal. Al crear un portal, su estado de publicación es `Disabled`. Esto significa que el portal no se puede encontrar en la web. Para que los consumidores accedan al portal, debe publicarlo. Antes de publicar el portal, también puede obtener una vista previa.

API Gateway admite portales en los siguientes navegadores:

- Firefox
- Google Chrome
- Microsoft Edge
- Safari

Elección de un nombre de dominio

De forma predeterminada, la URL del portal está alojada en el siguiente nombre de dominio propiedad de Amazon:

```
https://p-portalId.apigw-portal.us-east-1.on.aws
```

Para personalizar el nombre de dominio, puede proporcionar un nombre de dominio que le pertenezca y un certificado SSL para el dominio personalizado. El certificado se debe administrar con AWS Gestor de certificados (ACM) en Este de EE. UU. (Norte de Virginia) y debe ser compatible con Amazon CloudFront. Para obtener más información, consulte Requisitos para la utilización de certificados SSL/TLS con CloudFront.

Cuando utilice un nombre de dominio personalizado, API Gateway crea una distribución de CloudFront, protegida en tránsito con el certificado de ACM. Luego, API Gateway proporciona un registro de alias, que se debe agregar a la configuración de DNS. El registro de alias es el nombre de dominio propiedad de Amazon. Debe agregar este registro de alias para dirigir el tráfico a la distribución de CloudFront del dominio personalizado. El nombre de dominio que proporcione debe ser único y no estar utilizado ya en la lista de alias de una distribución de CloudFront existente.

Si usa su propio nombre de dominio, API Gateway usa la política de seguridad `TLS_1_2` para proteger el portal. Para obtener más información, consulte Protocolos y cifrados admitidos entre lectores. No puede cambiar este valor.

Creación de un portal

Utilice la consola de API Gateway para crear y configurar el portal. Los procedimientos siguientes lo guiarán a través de todos los pasos necesarios. También puede elegir Saltar para revisar y crear al final del primer procedimiento para crear un portal vacío.

El procedimiento siguiente muestra cómo crear un portal. Primero, defina los detalles del portal.

Definición de los detalles del portal

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Portales.
3. Elija Create portal (Crear portal).
4. En detalles del portal, para el Nombre del portal, ingrese un nombre.
5. Para Descripción del portal, ingrese una descripción.
6. Para la autorización del portal, seleccione cómo autorizar el portal.
   - Si es la primera vez que crea un portal, le recomendamos que elija Ninguno. Si tiene pensado agregar alguna API de producción al portal, utilice Requerir autorización.
   - Para exigir a los consumidores del portal que se autentiquen para usar el portal, elija Requerir autorización y haga lo siguiente:
     a. Para el grupo de usuarios de Amazon Cognito, seleccione su grupo de usuarios.

        El grupo de usuarios debe estar en la misma región que el portal.
     b. Para ID de cliente de la aplicación, elija el ID del cliente de la aplicación.

        En el cliente de la aplicación, debe establecer la URL de devolución de llamada en la URL predeterminada del portal.
     c. Para el dominio del grupo de usuarios, ingrese el dominio del grupo de usuarios. Para obtener más información, consulte Configuración de un dominio de grupo de usuarios.
7. Para la configuración del dominio, seleccione el nombre de dominio del portal.
   - Si es la primera vez que crea un portal, le recomendamos que elija Dominio predeterminado. Una vez que publique el portal, estará disponible en `https://p-portalId.apigw-portal.us-east-1.on.aws`.
   - Para proporcionar un nombre de dominio personalizado del que sea propietario, seleccione Nombre de dominio personalizado y haga lo siguiente:

a. Para Nombre de dominio, escriba un nombre de dominio completo.

b. Para Certificado de ACM, elija un certificado de ACM que cubra el nombre de dominio.

   Deberá agregar el registro de alias que API Gateway proporciona al registro de DNS para poder atender el tráfico del portal.

8. Para agregar un monitor de aplicaciones de CloudWatch RUM al portal, para el monitor de aplicaciones de RUM, elija un monitor de aplicaciones. Para obtener más información, consulte CloudWatch RUM.
9. Elija Siguiente.

Tras definir los detalles del portal, agregue los productos del portal. Puede cambiar los productos del portal en cualquier momento, pero tendrá que volver a publicar el portal para que los cambios surtan efecto.

Selección de los productos del portal

1. En la lista de recursos, seleccione los productos que desee agregar al portal.
2. Elija Siguiente.

Tras seleccionar los productos, personalice el diseño del portal. Para cargar un logotipo del portal, debe tener una imagen cargada en Amazon S3. Puede cambiar estos valores en cualquier momento, pero tendrá que volver a publicar el portal para que los cambios surtan efecto.

(Opcional) Personalización del portal

1. Para logotipo del portal, proporcione el URI de S3 del archivo.

   Solo `.svg`, `.png` o `.jpg` son compatibles.
2. Para personalizar el diseño del portal, tiene dos opciones:

   a. Puede elegir el tema del portal para aplicar un tema de color prediseñado al portal.

   b. Puede utilizar los selectores de color para seleccionar todas las opciones de diseño del portal.
3. Elija Siguiente.

Tras seleccionar el diseño del portal, revise la información del portal.

Revisión del portal

- Revise los ajustes en la página de revisión y creación. Cuando esté satisfecho con la configuración, elija Create (Crear).

Si utiliza un grupo de usuarios de Amazon Cognito para controlar el acceso al portal, debe establecer la URL de devolución de llamada en la URL predeterminada del portal en el cliente de la aplicación. Para obtener más información, consulte Application-specific settings with app clients.

El portal no estará disponible para los consumidores una vez que lo haya creado. Debe publicar el portal para que esté accesible en Internet. Para obtener más información, consulte the section called "Publicación de un portal".

Actualización de un portal en API Gateway

Puede actualizar el tema, el logotipo o los productos del portal en el portal. Al actualizar el portal, los cambios no surtirán efecto hasta que vuelva a publicar el portal.

Actualización de un portal

El siguiente procedimiento muestra cómo actualizar el diseño de un portal.

Actualización del diseño de un portal

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Portales.
3. Elija un portal.
4. En la pestaña Diseño del portal, elija Editar.
5. Para actualizar el logotipo del portal, elija Actualizar el logotipo del portal. Puede eliminar el logotipo existente o utilizar uno nuevo.
6. Realice cualquier actualización de tema o fondo en el diseño del portal.
7. Seleccione Save changes (Guardar cambios).
8. (Opcional) Elija Generar vista previa del portal para ver los cambios.

   Cuando haya confirmado los cambios, publique el portal.
9. Elija Publicar portal.

10. Ingrese una descripción para la actualización del portal y elija Publicar.

Vista previa de un portal en API Gateway

Puede obtener una vista previa del portal antes de publicarlo. API Gateway crea una URL de corta duración que puede usar para obtener una vista previa de la versión del portal que los clientes verán una vez publicado el portal.

Consideraciones

Las siguientes consideraciones pueden afectar a la vista previa de un portal:

- La prueba no está habilitada para el portal de vista previa. Para obtener más información, consulte the section called “Habilitación de una prueba para un punto de conexión REST de un producto en el portal”.

Vista previa de un portal

El siguiente procedimiento muestra cómo realizar una vista previa de un portal.

Realización de la vista previa de un portal

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Portales.
3. Elija un portal.
4. Elija Generar vista previa del portal.

   No actualice la página.
5. Una vez que API Gateway genere la vista previa, elija Abrir vista previa del portal.

   La vista previa del portal se abrirá en una pestaña nueva.

Publicación de un portal en API Gateway

Para que los consumidores de API accedan al portal, debe publicarlo. Cualquier persona en Internet puede detectar una URL de portal. Le recomendamos que obtenga una vista previa del portal y lo proteja antes de publicarlo.

Consideraciones

Es posible que API Gateway tarde unos minutos en publicar el portal. Puede supervisar el estado de publicación en la consola.

Publicación de un portal

El procedimiento siguiente muestra cómo publicar un portal.

Publicación de un portal

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Portales.
3. Elija un portal.
4. Elija Publicar portal.
5. (Opcional) En Descripción de los cambios, ingrese una descripción del cambio.

   Al publicar un portal, le recomendamos que proporcione siempre una breve descripción de los cambios.
6. Elija Publicar.

   Es posible que API Gateway tarde unos minutos en publicar el portal. API Gateway proporciona un enlace al portal cuando está disponible.

Antes de eliminar el portal, debe desactivarlo. Para obtener más información, consulte the section called “Desactivación de un portal”.

Uso de un portal de API Gateway

Después de publicar el portal, estará disponible en la web. Para usar el portal, copie el nombre de dominio predeterminado del portal o el nombre de dominio e ingréselo en un navegador. Si el portal usa un nombre de dominio, el nombre de dominio predeterminado no se puede usar para acceder al portal.

Uso de un portal

En el siguiente procedimiento, se muestra cómo acceder a un portal y usar la prueba para un punto de conexión REST del producto.

Uso de un portal

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Portales.
3. Elija un portal.
4. Copie el nombre de dominio predeterminado del portal o el nombre de dominio e ingréselo en un navegador.
5. En la página de inicio del portal, puede buscar productos del portal y detectar páginas de productos. Utilice la búsqueda en la parte superior del portal para buscar en todo el contenido del portal. Utilice la barra de búsqueda situada debajo de Productos para buscar los productos del portal.
6. Elija un producto del portal.
7. Elija un punto de conexión REST del producto.
8. En la ventana Pruébelo, ingrese los parámetros para intentar llamar a la API.
9. Elija Pruébelo. API Gateway invocará la API y mostrará los resultados en la ventana.

Solución de problemas del portal

A continuación, se le proporcionan consejos para solucionar errores y problemas que puedan surgir al utilizar un portal.

No se puede acceder al sitio de la página web del portal

Si recibe un mensaje de error que indica que no se puede acceder al sitio de la página web del portal, asegúrese de haber publicado el portal. Si usó un nombre de dominio, asegúrese de agregar el registro de alias que proporcionó API Gateway para dirigir el tráfico a la distribución de CloudFront para su dominio personalizado.

Prueba errónea

Si no puede usar la prueba para un punto de conexión REST del producto, asegúrese de que el estado de la prueba sea `ACTIVE`. Además, asegúrese de haber ingresado los parámetros correctos para invocar la API.

Las páginas no aparecen en el portal

Si ha creado una página de punto de conexión REST de un producto o una página de producto y no está visible en el portal, asegúrese de que las páginas no sean borradores. Para verificar esto, haga lo siguiente:

Comprobación de que las páginas no están en borradores

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Productos de portales.
3. Elija el producto del portal.
4. En la pestaña Documentación, confirme que las páginas no estén en la sección de borradores de páginas de documentación o borradores de páginas de referencia de la API, sino en las páginas de documentación personalizada o en las páginas de referencia de la API.

Desactivación de un portal en API Gateway

Al desactivar un portal, el portal deja de estar disponible en la web. Después de desactivar un portal, puede eliminarlo o volver a publicarlo si es necesario.

Desactivación de un portal

El procedimiento siguiente muestra cómo desactivar un portal.

Desactivación de un portal

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Portales.
3. Elija un portal.
4. Elija Acciones y, a continuación, Desactivar portal.
5. Confirme su elección y elija Desactivar portal.

   El estado de publicación cambiará a Desactivado.

Para eliminar el estado de desactivación del portal, vuelva a publicarlo.

Eliminación de un portal en API Gateway

Al eliminar un portal, no se puede recuperar y primero debe desactivarlo. Si desea eliminar el portal de la web, puede desactivarlo. Esto le permite modificar el portal y volver a publicarlo más adelante.

Eliminación de un portal

El procedimiento siguiente muestra cómo eliminar un portal.

Para eliminar un portal

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. En el panel de navegación principal, elija Portales.
3. Elija un portal.
4. Si el portal no está desactivado, desactívelo eligiendo Acciones, Desactivar portal.

   Se tarda unos minutos en desactivar el portal. Puede supervisar el estado de publicación para ver si el estado de publicación del portal está desactivado.
5. Una vez no publicado el portal, puede eliminarlo. Para eliminar el portal, elija Acciones, Eliminar portal. Confirme la elección y elija Eliminar.

API de HTTP de API Gateway

Las API de REST y las API HTTP son productos API de RESTful. Las API de REST admiten más funciones que las API HTTP, mientras que las API HTTP están diseñadas con características mínimas para que puedan ofrecerse a un precio más bajo. Para obtener más información, consulte the section called “Elección entre las API de REST y las API de HTTP ”.

Puede utilizar las API HTTP para enviar solicitudes a las funciones de AWS Lambda o a cualquier punto de enlace HTTP direccionable. Por ejemplo, puede crear una API HTTP que se integre con una función de Lambda en el backend. Cuando un cliente llama a la API, API Gateway envía la solicitud a la función de Lambda y devuelve la respuesta de la función al cliente.

Las API HTTP son compatibles con la autorización de OpenID Connect y OAuth 2.0. Vienen con soporte integrado para el uso compartido de recursos entre orígenes (CORS) y las implementaciones automáticas.

Puede crear API HTTP mediante la consola de administración de AWS, la AWS CLI, las API, CloudFormation o los SDK.

Temas

- Desarrollo de las API HTTP en API Gateway
- Publicación de las API de HTTP para que los clientes la invoquen
- Protección de sus API de HTTP en API Gateway
- Supervisión de las API de HTTP en API Gateway
- Solución de problemas con las API de HTTP en API Gateway

Desarrollo de las API HTTP en API Gateway

En esta sección se proporciona información detallada acerca de las capacidades de API Gateway que necesitará para desarrollar las API de API Gateway.

A medida que se desarrolla la API de API Gateway, se decide sobre una serie de características de la API. Estas características dependen del uso de la API. Por ejemplo, es posible que quiera permitir solo a ciertos clientes llamar a la API o puede que quiera que esté disponible para todos. Quizás desee que una llamada a la API ejecute una función de Lambda, haga una consulta a la base de datos o llame a una aplicación.

Temas

- Creación de una API de HTTP
- Creación de rutas para las API de HTTP en API Gateway
- Tipos de direcciones IP para las API HTTP en API Gateway
- Control y administración del acceso a las API HTTP en API Gateway
- Creación de integraciones para las API de HTTP en API Gateway
- Configuración de CORS de las API de HTTP en API Gateway
- Transformación de las solicitudes y respuestas de API para las API HTTP en API Gateway
- Uso de definiciones de OpenAPI para las API HTTP en API Gateway

Creación de una API de HTTP

Para crear una API funcional, debe contar al menos con una ruta, una integración, una etapa y una implementación.

En los siguientes ejemplos se muestra cómo crear una API con una integración de AWS Lambda o HTTP, una ruta y una etapa predeterminada configurada para implementar automáticamente los cambios.

En esta guía, se supone que ya está familiarizado con API Gateway y Lambda. Para obtener una guía más detallada, consulte Introducción.

Temas

- Crear una API HTTP mediante la Consola de administración de AWS
- Crear una API HTTP mediante la AWS CLI

Crear una API HTTP mediante la Consola de administración de AWS

1. Abra la consola de API Gateway.
2. Seleccione Create API (Crear API).
3. En HTTP API (API HTTP), elija Build (Compilación).
4. Elija Add integration (Añadir integración) y, a continuación, elija una función de AWS Lambda o escriba un punto de conexión HTTP.
5. En Name (Nombre), escriba un nombre para la API.

6. Elija Review and create (Revisar y crear).
7. Seleccione Create (Crear).

La API ya está lista para su invocación. Puede probar la API escribiendo su URL de invocación en un navegador o con Curl.

```
curl https://api-id.execute-api.us-east-2.amazonaws.com
```

Crear una API HTTP mediante la AWS CLI

Puede utilizar la creación rápida para crear una API con una integración de Lambda o HTTP, una ruta de método catch-all predeterminada y una etapa predeterminada configurada para implementar automáticamente los cambios. El siguiente comando create-api utiliza la creación rápida para crear una API que se integra con una función de Lambda en el backend.

> **Note**
>
> Para invocar una integración de Lambda, API Gateway debe tener los permisos necesarios. Puede utilizar una política basada en recursos o un rol de IAM para conceder a la API Gateway permisos para invocar una función de Lambda. Para obtener más información, consulte Permisos de AWS Lambda en la Guía del desarrollador de AWS Lambda.

Example

```
aws apigatewayv2 create-api --name my-api --protocol-type HTTP --target
 arn:aws:lambda:us-east-2:123456789012:function:function-name
```

La API ya está lista para su invocación. Puede probar la API escribiendo su URL de invocación en un navegador o con Curl.

```
curl https://api-id.execute-api.us-east-2.amazonaws.com
```

Creación de rutas para las API de HTTP en API Gateway

Las rutas dirigen las solicitudes entrantes de la API a los recursos de backend. Las rutas constan de dos partes: un método HTTP y una ruta de recurso, por ejemplo, `GET /pets`. Puede definir métodos

HTTP específicos para su ruta. O bien, puede utilizar el método ANY para que coincida con todos los métodos que no haya definido para un recurso. Puede crear una ruta `$default` que actúe como método catch-all para las solicitudes que no coincidan con ninguna otra ruta.

> **Note**
>
> API Gateway descodifica los parámetros de solicitud codificados con URL antes de pasarlos a las integraciones de backend.

Trabajar con variables de ruta

Puede utilizar variables de ruta en rutas de API HTTP.

Por ejemplo, la ruta `GET /pets/{petID}` captura una solicitud `GET` que un cliente envía a `https://api-id.execute-api.us-east-2.amazonaws.com/pets/6`.

Una variable de ruta ambiciosa captura todos los recursos secundarios de una ruta. Para crear una variable de ruta ambiciosa, agregue + al nombre de la variable, por ejemplo, `{proxy+}`. La variable de ruta expansiva debe estar al final de la ruta del recurso.

Trabajar con parámetros de cadena de consulta

De forma predeterminada, API Gateway envía parámetros de cadena de consulta a su integración de backend si se incluyen en una solicitud a una API HTTP.

Por ejemplo, cuando un cliente envía una solicitud a `https://api-id.execute-api.us-east-2.amazonaws.com/pets?id=4&type=dog`, los parámetros de cadena de consulta `?id=4&type=dog` se envían a su integración.

Trabajar con la ruta **`$default`**

La ruta `$default` captura solicitudes que no coinciden explícitamente con otras rutas en su API.

Cuando la ruta `$default` recibe una solicitud, API Gateway envía la ruta de solicitud completa a la integración. Por ejemplo, puede crear una API con solo una ruta `$default` e integrarla en el método ANY con el punto de enlace HTTP `https://petstore-demo-endpoint.execute-api.com`. Cuando envía una solicitud a `https://api-id.execute-api.us-east-2.amazonaws.com/store/checkout`, API Gateway envía una solicitud a `https://petstore-demo-endpoint.execute-api.com/store/checkout`.

Para obtener más información acerca de las integraciones de HTTP, consulte Uso de integraciones de proxy de HTTP para las API de HTTP.

Enrutamiento de solicitudes de la API

Cuando un cliente envía una solicitud de la API, API Gateway primero determina hacia qué etapa dirigir la solicitud. Si la solicitud coincide explícitamente con una etapa, API Gateway envía la solicitud a esa etapa. Si ninguna etapa coincide completamente con la solicitud, API Gateway envía la solicitud a la etapa `$default`. Si no hay ninguna etapa `$default`, la API devuelve `{"message":"Not Found"}` y no genera registros de CloudWatch.

Después de seleccionar una etapa, API Gateway selecciona una ruta. API Gateway selecciona la ruta con la coincidencia más específica y aplica las siguientes prioridades:

1. Coincidencia completa para una ruta y método.
2. Haga coincidir una ruta y un método con una variable de ruta ambiciosa (`{proxy+}`).
3. La ruta `$default`.

Si ninguna ruta coincide con una solicitud, API Gateway devuelve `{"message":"Not Found"}` al cliente.

Por ejemplo, piense en una API con una etapa `$default` y las siguientes rutas de ejemplo:

1. `GET /pets/dog/1`
2. `GET /pets/dog/{id}`
3. `GET /pets/{proxy+}`
4. `ANY /{proxy+}`
5. `$default`

   En la siguiente tabla se resume la forma en que API Gateway enruta las solicitudes a las rutas de ejemplo.

Solicitud	Ruta seleccionada	Explicación
`GET https://api-id.execute-`	`GET /pets/dog/1`	La solicitud coincide completamente con esta ruta estática.

Solicitud	Ruta seleccionada	Explicación
`api. region.amazonaws.com/pets/dog/1`		
`GET https://api-id.execute-api. region.amazonaws.com/pets/dog/2`	`GET /pets/dog/{id}`	La solicitud coincide completamente con esta ruta.
`GET https://api-id.execute-api. region.amazonaws.com/pets/cat/1`	`GET /pets/{proxy+}`	La solicitud no coincide completamente con una ruta. La ruta con un método `GET` y una variable de ruta ambiciosa captura esta solicitud.
`POST https://api-id.execute-api. region.amazonaws.com/test/5`	`ANY /{proxy+}`	El método `ANY` coincide con todos los métodos que no ha definido para una ruta. Las rutas con variables de ruta ambiciosas tienen mayor prioridad que la ruta `$default`.

Tipos de direcciones IP para las API HTTP en API Gateway

Al crear una API, se especifica el tipo de direcciones IP que puede invocar la API. Puede elegir IPv4 para resolver las direcciones IPv4 para invocar la API o puede elegir pila doble para permitir que las direcciones IPv4 e IPv6 invoquen la API. Es posible que desee configurar el tipo de dirección IP en pila doble para mitigar el agotamiento del espacio IP o para mejorar la postura de seguridad. Para obtener más información sobre los beneficios de un tipo de dirección IP de pila doble, consulte IPv6 en AWS.

Consideraciones para los tipos de direcciones IP

Es posible que las siguientes consideraciones afecten al uso de los tipos de direcciones IP:

- El tipo de dirección IP predeterminado para las API HTTP es IPv4.

- Si cambia el tipo de dirección IP de una API existente de IPv4 a pila doble, confirme que todas las políticas que controlan el acceso a las API se hayan actualizado para tener en cuenta las llamadas de IPv6. Cuando cambia el tipo de dirección IP, el cambio surte efecto inmediatamente.
- La API se puede asignar a un nombre de dominio personalizado con un tipo de dirección IP diferente al de la API. Si desactiva el punto de conexión de la API predeterminado, esto podría afectar a la forma en que los intermediarios pueden invocar a la API.

Cambio del tipo de dirección IP de una API HTTP

Puede cambiar el tipo de dirección IP actualizando la configuración de la API. Puede actualizar la configuración de la API mediante el uso de la Consola de administración de AWS, la AWS CLI, CloudFormation o un AWS SDK. Si cambia el tipo de dirección IP de la API, no es necesario volver a implementar la API para que los cambios surtan efecto.

Consola de administración de AWS

Cambio del tipo de dirección IP de una API HTTP

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Elija una API HTTP.
3. Para la configuración de la API, elija Editar.
4. Para tipo de dirección IP, seleccione IPv4 o Pila doble.
5. Seleccione Save.

   El cambio en la configuración de la API se aplicará de forma inmediata.

AWS CLI

El siguiente comando update-api actualiza una API para que tenga un tipo de dirección IP de pila doble:

```
aws apigatewayv2 update-api \
    --api-id abcd1234 \
    --ip-address-type dualstack
```

El resultado será similar al siguiente:

```
{
```

```
    "ApiEndpoint": "https://abcd1234.execute-api.us-east-1.amazonaws.com",
    "ApiId": "abcd1234",
    "ApiKeySelectionExpression": "$request.header.x-api-key",
    "CreatedDate": "2025-02-04T22:20:20+00:00",
    "DisableExecuteApiEndpoint": false,
    "Name": "My-HTTP-API",
    "ProtocolType": "HTTP",
    "RouteSelectionExpression": "$request.method $request.path",
    "Tags": {},
    "NotificationUris": [],
    "IpAddressType": "dualstack"
}
```


Control y administración del acceso a las API HTTP en API Gateway

API Gateway admite varios mecanismos para controlar y administrar el acceso a la API HTTP:

- Los autorizadores de Lambda utilizan las funciones de Lambda para controlar el acceso a las API. Para obtener más información, consulte Control del acceso a las API de HTTP con autorizadores de AWS Lambda.
- Los autorizadores de JWT utilizan tokens web JSON para controlar el acceso a las API. Para obtener más información, consulte Control del acceso a API HTTP con autorizadores de JWT en API Gateway.
- Los roles y las políticas estándar de AWS IAM ofrecen controles de acceso flexibles y robustos. Puede usar roles y políticas de IAM para controlar quién puede crear y administrar sus API, así como quién puede invocarlas. Para obtener más información, consulte Control del acceso a las API de HTTP con autorización de IAM en API Gateway.

Para aumentar la seguridad, le recomendamos que configure un autorizador para todas las rutas de su API HTTP. Es probable que deba hacerlo para asegurar el cumplimiento con diversos marcos normativos. Para obtener más información, consulte Controles de Amazon API Gateway en la Guía del usuario de AWS Security Hub.

Control del acceso a las API de HTTP con autorizadores de AWS Lambda

El autorizador de Lambda se utiliza para emplear una función de Lambda para controlar el acceso a una API HTTP. Después, cuando un cliente llama a dicha API, API Gateway invoca la función de

Lambda. API Gateway utiliza la respuesta de su función de Lambda para determinar si el cliente puede acceder a la API.

Versión de formato de carga

La versión del formato de carga del autorizador especifica el formato de los datos que API Gateway envía a un autorizador de Lambda y cómo API Gateway interpreta la respuesta de Lambda. Si no especifica una versión de formato de carga, la Consola de administración de AWS utiliza la versión más reciente de forma predeterminada. Si crea un autorizador de Lambda mediante la AWS CLI, CloudFormation o un SDK, debe especificar una `authorizerPayloadFormatVersion`. Los valores admitidos son `1.0` y `2.0`.

Si necesita compatibilidad con las API de REST, utilice la versión `1.0`.

Los siguientes ejemplos muestran la estructura de cada versión de formato de carga.

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"
  }
}
```

Formato de respuesta del autorizador de Lambda

La versión del formato de carga también determina la estructura de la respuesta que debe devolver de su función de Lambda.

Respuesta de función de Lambda para el formato 1.0

Si elige la versión de formato `1.0`, los autorizadores de Lambda deben devolver una política de IAM que permita o deniegue el acceso a su ruta de la API. Puede utilizar la sintaxis de política de IAM estándar en la política. Para obtener algunos ejemplos de políticas de IAM, consulte the section called “Controlar el acceso para invocar una API”. Puede transmitir propiedades de contexto a integraciones o registros de acceso de Lambda mediante `$context.authorizer.property`. El objeto `context` es opcional y `claims` es un marcador de posición reservado y no se puede utilizar como objeto de contexto. Para obtener más información, consulte the section called “Variables de registro”.

Example

JSON

```
{
  "principalId": "abcdef",
  "policyDocument": {
    "Version":"2012-10-17",
    "Statement": [
      {
        "Action": "execute-api:Invoke",
        "Effect": "Allow|Deny",
        "Resource": "arn:aws:execute-api:{regionId}:{accountId}:{apiId}/{stage}/
{httpVerb}/[{resource}/[{child-resources}]]"
      }
    ]
  },
  "context": {
    "exampleKey": "exampleValue"
  }
}
```

Respuesta de función de Lambda para el formato 2.0

Si elige la versión de formato `2.0`, puede devolver un valor booleano o una política de IAM que utilice la sintaxis de la política de IAM estándar de la función de Lambda. Para devolver un valor booleano, habilite respuestas simples para el autorizador. Los siguientes ejemplos muestran el formato que debe utilizar para codificar la devolución de la función de Lambda. El objeto `context` es opcional. Puede transmitir propiedades de contexto a integraciones o registros de acceso de Lambda mediante `$context.authorizer.property`. Para obtener más información, consulte the section called “Variables de registro”.

Simple response

```
{
  "isAuthorized": true/false,
  "context": {
    "exampleKey": "exampleValue"
  }
}
```

IAM policy

```
{
  "principalId": "abcdef",
  "policyDocument": {
    "Version":"2012-10-17",
    "Statement": [
      {
        "Action": "execute-api:Invoke",
        "Effect": "Allow|Deny",
        "Resource": "arn:aws:execute-api:{regionId}:{accountId}:{apiId}/{stage}/
{httpVerb}/[{resource}/[{child-resources}]]"
      }
    ]
  },
  "context": {
    "exampleKey": "exampleValue"
  }
}
```

Ejemplo de funciones de autorizador de Lambda

El siguiente ejemplo de funciones de Lambda Node.js muestran los formatos de respuesta necesarios que debe devolver desde la función de Lambda para la versión del formato de carga `2.0`.

Simple response - Node.js

```
export const handler = async(event) => {
    let response = {
        "isAuthorized": false,
        "context": {
            "stringKey": "value",
            "numberKey": 1,
            "booleanKey": true,
            "arrayKey": ["value1", "value2"],
            "mapKey": {"value1": "value2"}
        }
    };

    if (event.headers.authorization === "secretToken") {
        console.log("allowed");
        response = {
            "isAuthorized": true,
            "context": {
                "stringKey": "value",
                "numberKey": 1,
                "booleanKey": true,
                "arrayKey": ["value1", "value2"],
                "mapKey": {"value1": "value2"}
            }
        };
    }

    return response;

};
```

Simple response - Python

```
import json


def lambda_handler(event, context):
```

```
    response = {
        "isAuthorized": False,
        "context": {
            "stringKey": "value",
            "numberKey": 1,
            "booleanKey": True,
            "arrayKey": ["value1", "value2"],
            "mapKey": {"value1": "value2"}
        }
    }

    try:
        if (event["headers"]["authorization"] == "secretToken"):
            response = {
                "isAuthorized": True,
                "context": {
                    "stringKey": "value",
                    "numberKey": 1,
                    "booleanKey": True,
                    "arrayKey": ["value1", "value2"],
                    "mapKey": {"value1": "value2"}
                }
            }
            print('allowed')
            return response
        else:
            print('denied')
            return response
    except BaseException:
        print('denied')
        return response
```

IAM policy - Node.js

```
export const handler = async(event) => {
  if (event.headers.authorization == "secretToken") {
    console.log("allowed");
    return {
      "principalId": "abcdef", // The principal user identification associated with
 the token sent by the client.
      "policyDocument": {
        "Version": "2012-10-17",
        "Statement": [{
```

```
          "Action": "execute-api:Invoke",
          "Effect": "Allow",
          "Resource": event.routeArn
        }]
      },
      "context": {
        "stringKey": "value",
        "numberKey": 1,
        "booleanKey": true,
        "arrayKey": ["value1", "value2"],
        "mapKey": { "value1": "value2" }
      }
    };
  }
  else {
    console.log("denied");
    return {
      "principalId": "abcdef", // The principal user identification associated with
 the token sent by the client.
      "policyDocument": {
        "Version": "2012-10-17",
        "Statement": [{
          "Action": "execute-api:Invoke",
          "Effect": "Deny",
          "Resource": event.routeArn
        }]
      },
      "context": {
        "stringKey": "value",
        "numberKey": 1,
        "booleanKey": true,
        "arrayKey": ["value1", "value2"],
        "mapKey": { "value1": "value2" }
      }
    };
  }
};
```

IAM policy - Python

```
import json
```

```
def lambda_handler(event, context):
    response = {
        # The principal user identification associated with the token sent by
        # the client.
        "principalId": "abcdef",
        "policyDocument": {
            "Version": "2012-10-17",
            "Statement": [{
                "Action": "execute-api:Invoke",
                "Effect": "Deny",
                "Resource": event["routeArn"]
            }]
        },
        "context": {
            "stringKey": "value",
            "numberKey": 1,
            "booleanKey": True,
            "arrayKey": ["value1", "value2"],
            "mapKey": {"value1": "value2"}
        }
    }

    try:
        if (event["headers"]["authorization"] == "secretToken"):
            response = {
                # The principal user identification associated with the token
                # sent by the client.
                "principalId": "abcdef",
                "policyDocument": {
                    "Version": "2012-10-17",
                    "Statement": [{
                        "Action": "execute-api:Invoke",
                        "Effect": "Allow",
                        "Resource": event["routeArn"]
                    }]
                },
                "context": {
                    "stringKey": "value",
                    "numberKey": 1,
                    "booleanKey": True,
                    "arrayKey": ["value1", "value2"],
                    "mapKey": {"value1": "value2"}
                }
            }
```

```
            print('allowed')
            return response
        else:
            print('denied')
            return response
    except BaseException:
        print('denied')
        return response
```

Fuentes de identidad

Opcionalmente, puede especificar orígenes de identidad para un autorizador de Lambda. Los orígenes de identidad especifican la ubicación de los datos necesarios para autorizar una solicitud. Por ejemplo, puede especificar valores de cabecera o cadena de consulta como orígenes de identidad. Si especifica orígenes de identidad, los clientes deben incluirlos en la solicitud. Si la solicitud del cliente no incluye los orígenes de identidad, API Gateway no invoca el autorizador de Lambda y el cliente recibe un error `401`.

En la siguiente tabla se describen los orígenes de identidad admitidos para un autorizador de Lambda.

Tipo	Ejemplo	Notas
Valor del encabezado	$request.header.*name*	Los nombres del encabezado no distinguen entre mayúsculas y minúsculas.
Valor de la cadena de consulta	$request.querystring.*name*	Los nombres de las cadenas de consulta distinguen mayúsculas y minúsculas.
Variable de contexto	$contexto.*variableName*	El valor de una variable de contexto compatible.
Variable de etapa	$stageVariables.*variableName*	El valor de una variable de etapa.

También puede devolver directamente `{"errorMessage" : "Unauthorized"}` desde la función de Lambda para devolver un error `401` a los clientes. Si devuelve directamente un error `401` desde la función de Lambda a los clientes, no especifique ninguna fuente de identidad al crear el autorizador de Lambda.

Almacenamiento en caché de las respuestas de los autorizadores

Para habilitar el almacenamiento en caché de un autorizador de Lambda debe especificar un authorizerResultTtlInSeconds. Cuando el almacenamiento en caché está habilitado para un autorizador, API Gateway utiliza los orígenes de identidad del autorizador como clave de caché. Si un cliente especifica los mismos parámetros en orígenes de identidad dentro del TTL configurado, API Gateway utiliza el resultado del autorizador almacenado en caché, en lugar de invocar su función de Lambda.

Para habilitar el almacenamiento en caché, el autorizador debe tener al menos un origen de identidad.

Si habilita respuestas simples para un autorizador, la respuesta del autorizador permite o deniega completamente todas las solicitudes de la API que coincidan con los valores de origen de identidad almacenados en caché. Para obtener permisos pormenorizados, desactive las respuestas simples y devuelva una política de IAM. En función del autorizador, es posible que la política de IAM necesite controlar el acceso a múltiples.

De forma predeterminada, API Gateway utiliza la respuesta del autorizador almacenada en caché para todas las rutas de una API que utiliza el autorizador. Para almacenar en caché las respuestas por ruta, agregue `$context.routeKey` a las fuentes de identidad del autorizador.

Crear un autorizador de Lambda

Al crear un autorizador de Lambda, se especifica la función de Lambda que utilizará API Gateway. Debe conceder permiso de API Gateway para invocar la función de Lambda mediante la política de recursos de la función o un rol de IAM. El siguiente comando create-authorizer permite crear un autorizador de Lambda:

```
aws apigatewayv2 create-authorizer \
    --api-id abcdef123 \
    --authorizer-type REQUEST \
    --identity-source '$request.header.Authorization' \
    --name lambda-authorizer \
    --authorizer-uri 'arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-west-2:123456789012:function:my-function/invocations' \
```

```
    --authorizer-payload-format-version '2.0' \
    --enable-simple-responses
```

El siguiente comando add-permission permite actualizar la política de recursos de la función de Lambda para conceder a API Gateway permiso para invocar la función. Si API Gateway no tiene permiso para invocar la función, los clientes reciben un `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"
```

Una vez se haya creado un autorizador y concedido permiso a API Gateway para invocarlo, actualice la ruta para poder utilizarlo. El siguiente comando update-route permite agregar el autorizador de Lambda a la ruta. Si el autorizador de Lambda utiliza el almacenamiento en caché de la política, asegúrese de actualizar la política para controlar el acceso de la ruta adicional.

```
aws apigatewayv2 update-route \
    --api-id abcdef123 \
    --route-id abc123 \
    --authorization-type CUSTOM \
    --authorizer-id def123
```

Resolución de problemas de los autorizadores de Lambda

Si API Gateway no puede invocar su autorizador de Lambda, o su autorizador de Lambda devuelve una respuesta en un formato no válido, los clientes recibirán un `500 Internal Server Error`.

Para solucionar errores, habilite el registro de acceso en la etapa de la API. Incluya la variable de registro `$context.authorizer.error` en su formato de registro.

Si los registros indican que API Gateway no tiene permiso para invocar la función, actualice la política de recursos de la función o proporcione un rol de IAM para conceder permiso a API Gateway para invocar el autorizador.

Si los registros indican que la función de Lambda devuelve una respuesta no válida, compruebe que la función de Lambda devuelve una respuesta en el formato requerido.

Control del acceso a API HTTP con autorizadores de JWT en API Gateway

Puede utilizar tokens web JSON (JWT) como parte de los marcos OpenID Connect (OIDC) y OAuth 2.0 para restringir el acceso de clientes a las API.

Si configura un autorizador de JWT para una ruta de la API, API Gateway valida los JWT que los clientes envían con solicitudes de la API. API Gateway permite o deniega las solicitudes en función de la validación del token y, opcionalmente, de los ámbitos del token. Si configura ámbitos para una ruta, el token debe incluir al menos uno de los ámbitos de la ruta.

Puede configurar distintos autorizadores para cada ruta de una API o utilizar el mismo autorizador para varias rutas.

> **Note**
>
> No existe ningún mecanismo estándar para diferenciar los tokens de acceso de JWT de otros tipos de JWT, como los tokens de ID de OpenID Connect. A menos que necesite tokens de ID para la autorización de la API, le recomendamos que configure las rutas para que soliciten los ámbitos de la autorización. También puede configurar los autorizadores de JWT para que soliciten los emisores o los destinatarios que utiliza el proveedor de identidades solo al emitir tokens de acceso de JWT.

Autorización de solicitudes de API con un autorizador de JWT

API Gateway utiliza el siguiente flujo de trabajo general para autorizar solicitudes a rutas configuradas para utilizar un autorizador de JWT.

1. Comprueba si `identitySource` contiene un token. `identitySource` puede incluir solo el token o el token con el prefijo `Bearer`.
2. Descodifique el token.
3. Compruebe el algoritmo y la firma del token tilizando la clave pública obtenida del del emiso `jwks_uri`. Actualmente solo se admiten algoritmos basados en RSA. API Gateway almacena en caché la clave pública durante un máximo de dos horas en la medida de lo posible. En caso de error de la caché, API Gateway obtiene la clave pública del emisor a través de la red, lo que agrega latencia a la solicitud que la desencadena. Es más probable que las API con una tasa de solicitudes baja o intermitente experimenten estas recuperaciones. Dado que una clave pública puede permanecer en caché hasta dos horas, al rotar las claves se recomienda dejar un periodo de gracia durante el cual tanto la clave antigua como la nueva sean válidas.

4. Valide las reclamaciones. API Gateway evalúa las siguientes reclamaciones de token:
   - `kid`: el token debe tener una reclamación de encabezado que coincida con la clave de `jwks_uri` que firmó el token.
   - `iss`: debe coincidir con el `issuer` configurado para el autorizador.
   - `aud` o `client_id`: deben coincidir con una de las entradas de `audience` configuradas para el autorizador. API Gateway valida `client_id` solo si `aud` no está presente. Cuando `aud` y `client_id` están presentes, API Gateway evalúa `aud`.
   - `exp`: debe ser posterior a la hora actual en UTC.
   - `nbf`: debe ser anterior a la hora actual en UTC.
   - `iat`: debe ser anterior a la hora actual en UTC.
   - `scope` o `scp`: el token debe incluir al menos uno de los ámbitos de los `authorizationScopes` de la ruta.

Si falla alguno de estos pasos, API Gateway deniega la solicitud de la API.

Después de validar el JWT, API Gateway transfiere las reclamaciones del token a la integración de la ruta de la API. Los recursos de backend, como las funciones de Lambda, pueden acceder a las reclamaciones de JWT. Por ejemplo, si el JWT incluye una reclamación de identidad `emailID`, esta estará disponible para una integración de Lambda en `$event.requestContext.authorizer.jwt.claims.emailID`. Para obtener más información acerca de la carga que API Gateway envía a las integraciones de Lambda, consulte the section called “AWS LambdaIntegraciones de ”.

Crear un autorizador de JWT

Antes de crear un autorizador de JWT, debe registrar una aplicación cliente con un proveedor de identidad. También debe haber creado una API HTTP. Para obtener ejemplos de creación de una API HTTP, consulte Creación de una API de HTTP.

Creación de un autorizador de JWT a través de la consola

Los siguientes pasos muestran cómo crear un autorizador de JWT mediante la consola.

Creación de un autorizador de JWT a través de la consola

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Elija una API HTTP.

3. En el panel de navegación principal, elija Autorización.
4. Elija la pestaña Administrar autorizadores.
5. Seleccione Crear.
6. Para Tipo de autorizador, elija JWT.
7. Configure el autorizador de JWT y especifique un Origen de identidad que defina el origen del token.
8. Seleccione Crear.

Creación de un autorizador de JWT con la AWS CLI

El siguiente comando create-authorizer crea un autorizador de JWT. Para `jwt-configuration`, especifique la `Audience` y el `Issuer` del proveedor de identidades. Si utiliza Amazon Cognito como proveedor de identidades, `IssuerUrl` es `https://cognito-idp.`*`us-east-2`*`.amazonaws.com/`*`userPoolID`*.

```
aws apigatewayv2 create-authorizer \
    --name authorizer-name \
    --api-id api-id \
    --authorizer-type JWT \
    --identity-source '$request.header.Authorization' \
    --jwt-configuration Audience=audience,Issuer=IssuerUrl
```

Creación de un autorizador de JWT a través de AWS CloudFormation

La siguiente plantilla de CloudFormation crea una API HTTP con un autorizador de JWT que usa Amazon Cognito como proveedor de identidades.

El resultado de la plantilla de CloudFormation es una URL de una interfaz de usuario alojada de Amazon Cognito en la que los clientes pueden registrarse e iniciar sesión para recibir un JWT. Cuando un cliente inicia sesión, se le redirige a la API HTTP con un token de acceso en la URL. Para invocar la API con el token de acceso, cambie # en la URL por ? para usar el token como un parámetro de cadena de consulta.

Plantilla de CloudFormation de ejemplo

```
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
```

Actualización de una ruta para utilizar un autorizador de JWT

Puede usar la consola, la AWS CLI, o un AWS SDK para actualizar una ruta y usar un autorizador de JWT.

Actualización de una ruta para utilizar un autorizador de JWT a través de la consola

Los siguientes pasos muestran cómo actualizar una ruta para usar un autorizador de JWT mediante la consola.

Creación de un autorizador de JWT a través de la consola

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Elija una API HTTP.
3. En el panel de navegación principal, elija Autorización.
4. Elija un método y, a continuación, seleccione el autorizador del menú desplegable y elija Asociar autorizador.

Actualización de una ruta para utilizar un autorizador de JWT a través de la AWS CLI

El siguiente comando update-route actualiza una ruta para que utilice un autorizador de JWT.

```
aws apigatewayv2 update-route \
   --api-id api-id  \
   --route-id route-id  \
   --authorization-type JWT \
   --authorizer-id authorizer-id \
   --authorization-scopes user.email
```

Control del acceso a las API de HTTP con autorización de IAM en API Gateway

Puede habilitar la autorización de IAM para rutas de API HTTP. Cuando la autorización de IAM está habilitada, los clientes deben utilizar Signature Version 4 (SigV4) para firmar las solicitudes con credenciales de AWS. API Gateway invoca la ruta de su API solo si el cliente dispone del permiso `execute-api` para la ruta.

La autorización de IAM para las API HTTP es similar a la de las API de REST.

> **Note**
>
> Actualmente, las políticas de recursos no se admiten para las API HTTP.

Para obtener ejemplos de políticas de IAM que otorgan a los clientes el permiso para invocar las API, consulte the section called “Controlar el acceso para invocar una API”.

Habilitación de la autorización de IAM para una ruta

El siguiente comando update-route habilita la autorización de IAM para una ruta de API HTTP.

```
aws apigatewayv2 update-route \
    --api-id abc123 \
    --route-id abcdef \
    --authorization-type AWS_IAM
```

Creación de integraciones para las API de HTTP en API Gateway

Las integraciones conectan una ruta a los recursos de backend. Las API HTTP admiten integraciones de proxy de Lambda, servicio de AWS y proxy de HTTP. Por ejemplo, puede configurar una solicitud `POST` para la ruta `/signup` de la API para que se integre con una función de Lambda que se encarga del registro de clientes.

Temas

- Creación de integraciones de proxy de AWS Lambda para las API de HTTP en API Gateway
- Uso de integraciones de proxy de HTTP para las API de HTTP
- Creación de integraciones de servicio de AWS para las API de HTTP en API Gateway
- Creación de integraciones privadas para API HTTP en API Gateway

Creación de integraciones de proxy de AWS Lambda para las API de HTTP en API Gateway

Una integración de proxy de Lambda le permite integrar una ruta de API con una función de Lambda. Cuando un cliente llama a la API, API Gateway envía la solicitud a la función de Lambda y devuelve la respuesta de la función al cliente. Para obtener ejemplos de creación de una API HTTP, consulte Creación de una API de HTTP.

Versión de formato de carga

La versión del formato de carga especifica el formato del evento que API Gateway envía a una integración de Lambda y cómo API Gateway interpreta la respuesta de Lambda. Si no especifica una versión de formato de carga, la Consola de administración de AWS utiliza la versión más reciente de forma predeterminada. Si crea una integración de Lambda utilizando la AWS CLI, CloudFormation o un SDK, debe especificar una `payloadFormatVersion`. Los valores admitidos son `1.0` y `2.0`.

Para obtener más información sobre cómo establecer `payloadFormatVersion`, consulte create-integration. Para obtener más información sobre cómo determinar la `payloadFormatVersion` de una integración existente, consulte get-integration.

Diferencias de formato de carga

En la siguiente lista, se muestran las diferencias entre las versiones del formato de carga `1.0` y `2.0`:

- El formato `2.0` no tiene campos `multiValueHeaders` o `multiValueQueryStringParameters`. Los encabezados duplicados se combinan con comas y se incluyen en el campo `headers`. Las cadenas de consulta duplicadas se combinan con comas y se incluyen en el campo `queryStringParameters`.
- El formato `2.0` tiene `rawPath`. Si utiliza una asignación de API para conectar su escenario a un nombre de dominio personalizado, `rawPath` no proporcionará el valor de asignación de la API. Use el formato `1.0` y `path` para acceder a la asignación de la API para su nombre de dominio personalizado.
- El formato `2.0` incluye un nuevo campo `cookies`. Todos los encabezados de cookies en la solicitud se combinan con comas y se agregan al campo `cookies`. En la respuesta al cliente, cada cookie se convierte en un encabezado `set-cookie`.

Estructura de formato de carga

Los siguientes ejemplos muestran la estructura de cada versión de formato de carga. Todos los nombres de encabezado están en minúsculas.

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
}
```

Formato de respuesta de función de Lambda

La versión del formato de carga determina la estructura de la respuesta que debe devolver su función de Lambda.

Respuesta de función de Lambda para el formato 1.0

Con la versión de formato `1.0`, las integraciones de Lambda deben devolver una respuesta en el siguiente formato JSON:

Example

```
{
    "isBase64Encoded": true|false,
    "statusCode": httpStatusCode,
    "headers": { "headername": "headervalue", ... },
    "multiValueHeaders": { "headername": ["headervalue", "headervalue2", ...], ... },
    "body": "..."
}
```

Respuesta de función de Lambda para el formato 2.0

Con la versión de formato `2.0`, API Gateway puede inferir el formato de respuesta por usted. API Gateway hace las siguientes suposiciones si su función de Lambda devuelve JSON válido y no devuelve un `statusCode`:

- `isBase64Encoded` es `false`.
- `statusCode` es `200`.
- `content-type` es `application/json`.
- `body` es la respuesta de la función.

Los siguientes ejemplos muestran el resultado de una función de Lambda y la interpretación de API Gateway.

Salida de función de Lambda	Interpretación de API Gateway
`"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" } }`

Para personalizar la respuesta, su función de Lambda debe devolver una respuesta con el siguiente formato.

```
{
    "cookies" : ["cookie1", "cookie2"],
    "isBase64Encoded": true|false,
    "statusCode": httpStatusCode,
    "headers": { "headername": "headervalue", ... },
    "body": "Hello from Lambda!"
}
```

Uso de integraciones de proxy de HTTP para las API de HTTP

Una integración de proxy HTTP le permite conectar una ruta de API a un punto de enlace HTTP enrutable públicamente. Con este tipo de integración, API Gateway pasa toda la solicitud y respuesta entre el frontend y el backend.

Para crear una integración de proxy HTTP, proporcione la dirección URL de un punto de enlace HTTP direccionable públicamente.

Integración de proxy HTTP con variables de ruta

Puede utilizar variables de ruta en rutas de API HTTP.

Por ejemplo, la ruta `/pets/{petID}` captura solicitudes a `/pets/6`. Puede hacer referencia a variables de ruta en el URI de integración para enviar el contenido de la variable a una integración. Un ejemplo es: `/pets/extendedpath/{petID}`.

Puede usar variables de ruta ambiciosas para capturar todos los recursos secundarios de una ruta. Para crear una variable de ruta ambiciosa, agregue + al nombre de la variable, por ejemplo, `{proxy+}`.

Para configurar una ruta con una integración de proxy HTTP que capte todas las solicitudes, cree una ruta API con una variable de ruta ambiciosa (por ejemplo, `/parent/{proxy+}`). Integre la ruta con un punto de enlace HTTP (por ejemplo, `https://petstore-demo-endpoint.execute-api.com/petstore/{proxy}`) en el método `ANY`. La variable de ruta expansiva debe estar al final de la ruta del recurso.

Creación de integraciones de servicio de AWS para las API de HTTP en API Gateway

Puede integrar su API HTTP en los servicios de AWS usando integraciones de primera clase. La integración de primera clase conecta una ruta de API HTTP a una API de servicio de AWS. Cuando un cliente invoca una ruta respaldada por una integración de primera clase, API Gateway invoca una API de servicio de AWS por usted. Por ejemplo, puede utilizar integraciones de primera clase para enviar un mensaje a una cola de Amazon Simple Queue Service o para iniciar una máquina de estado de AWS Step Functions. Para ver las acciones de servicio compatibles, consulte the section called “AWSReferencia de integraciones de servicios de ”.

Asignar parámetros de solicitud

Las integraciones de primera clase tienen parámetros obligatorios y opcionales. Debe configurar todos los parámetros que son obligatorios para crear una integración. Puede utilizar valores estáticos o parámetros de asignación que se evalúan dinámicamente en el tiempo de ejecución. Para obtener una lista completa de las integraciones y parámetros admitidos, consulte the section called “AWSReferencia de integraciones de servicios de ”.

En la siguiente tabla, se describen los parámetros de solicitud de asignación admitidos.

Tipo	Ejemplo	Notas
Valor del encabezado	$request.header.*name*	Los nombres del encabezado no distinguen entre mayúsculas y minúsculas. API Gateway combina varios valores de encabezado con comas, por ejemplo, `"header1": "value1,value2"` .
Valor de la cadena de consulta	$request.querystring.*name*	Los nombres de las cadenas de consulta distinguen mayúsculas y minúsculas. API Gateway combina varios valores con comas, por ejemplo, `"querystring1": "Value1,Value2"` .
Parámetro de ruta	$request.path.*name*	El valor de un parámetro de ruta en la solicitud. Por ejemplo, si la ruta es `/pets/{petId}` , puede asignar el parámetro `petId` de la solicitud con *`$request.path.petId`* .
Solicitar acceso directo del cuerpo	$request.body	API Gateway transfiere todo el cuerpo de la solicitud directamente.
Cuerpo de la solicitud	$request.body.*name*	Una expresión de ruta de acceso JSON. El descenso recursivo (`$request.body..` *`name`*) y las expresiones de filtro (`?(`*`expression`*`)`) no son compatibles.

Tipo	Ejemplo	Notas
		Note Cuando se especific a una ruta JSON, API Gateway trunca el cuerpo de la solicitud en 100 KB y, a continuación, aplica la expresión de selección. Para enviar cargas de más de 100 KB, especifiq ue `$request.body` .
Variable de contexto	$contexto.*variableName*	El valor de una variable de contexto compatible.
Variable de etapa	$stageVariables.*variableName*	El valor de una variable de etapa.
Valor estático	*cadena*	Un valor constante.

Crear una integración de primera clase

Antes de crear una integración de primera clase, debe crear un rol de IAM que conceda a API Gateway permisos para invocar la acción de servicio de AWS en la que se está integrando. Para obtener más información, consulte Creación de un rol para un servicio de AWS.

Para crear una integración de primera clase, elija una acción de servicio de AWS admitido, como `SQS-SendMessage`, configure los parámetros de la solicitud y proporcione un rol que conceda a API Gateway permisos para invocar la API de servicio de AWS integrada. Dependiendo del subtipo de integración, se requieren diferentes parámetros de solicitud. Para obtener más información, consulte the section called “AWSReferencia de integraciones de servicios de ”.

El siguiente comando create-integration permite crear una integración que envía un mensaje de Amazon SQS.

```
aws apigatewayv2 create-integration \
    --api-id abcdef123 \
    --integration-subtype SQS-SendMessage \
    --integration-type AWS_PROXY \
    --payload-format-version 1.0 \
    --credentials-arn arn:aws:iam::123456789012:role/apigateway-sqs \
    --request-parameters '{"QueueUrl": "$request.header.queueUrl", "MessageBody":
 "$request.body.message"}'
```

Creación de una integración de primera clase con CloudFormation

El siguiente ejemplo muestra un fragmento de CloudFormation que crea una ruta de `/{source}/{detailType}` con una integración de primera clase con Amazon EventBridge.

El parámetro `Source` se asigna al parámetro de ruta `{source}`, el `DetailType` se asigna al parámetro de ruta `{DetailType}` y el parámetro `Detail` se asigna al cuerpo de la solicitud.

El fragmento no muestra el bus de eventos ni el rol de IAM que otorga permisos a API Gateway para invocar la acción `PutEvents`.

```
Route:
    Type: AWS::ApiGatewayV2::Route
    Properties:
      ApiId: !Ref HttpApi
      AuthorizationType: None
      RouteKey: 'POST /{source}/{detailType}'
      Target: !Join
        - /
        - - integrations
          - !Ref Integration
  Integration:
    Type: AWS::ApiGatewayV2::Integration
    Properties:
      ApiId: !Ref HttpApi
      IntegrationType: AWS_PROXY
      IntegrationSubtype: EventBridge-PutEvents
      CredentialsArn: !GetAtt EventBridgeRole.Arn
      RequestParameters:
        Source: $request.path.source
        DetailType: $request.path.detailType
        Detail: $request.body
        EventBusName: !GetAtt EventBus.Arn
```

```
      PayloadFormatVersion: "1.0"
```

Referencia del subtipo de integración

Los siguientes subtipos de integración son compatibles con las API HTTP.

Subtipos de integración

- EventBridge-PutEvents 1.0
- SQS-SendMessage 1.0
- SQS-ReceiveMessage 1.0
- SQS-DeleteMessage 1.0
- SQS-PurgeQueue 1.0
- AppConfig-GetConfiguration 1.0
- Kinesis-PutRecord 1.0
- StepFunctions-StartExecution 1.0
- StepFunctions: StartSyncExecution 1.0
- StepFunctions-StopExecution 1.0

EventBridge-PutEvents 1.0

Envía eventos personalizados a Amazon EventBridge para que puedan ajustarse a las reglas.

Parámetro	Obligatorio
Detalle	True
DetailType	True
Fuente	True
Time	Falso
EventBusName	Falso
Recursos	Falso
Región	Falso

Parámetro	Obligatorio
TraceHeader	Falso

Para obtener más información, consulte PutEvents en la Referencia de la API de Amazon EventBridge.

SQS-SendMessage 1.0

Entrega un mensaje a la cola especificada.

Parámetro	Obligatorio
QueueUrl	True
MessageBody	True
DelaySeconds	Falso
MessageAttributes	Falso
MessageDeduplicationId	Falso
MessageGroupId	Falso
MessageSystemAttributes	Falso
Región	Falso

Para obtener más información, consulte SendMessage en la Referencia de la API de Amazon Simple Queue Service.

SQS-ReceiveMessage 1.0

Recupera uno o varios mensajes (hasta 10) de la cola especificada.

Parámetro	Obligatorio
QueueUrl	True

Parámetro	Obligatorio
AttributeNames	Falso
MaxNumberOfMessages	Falso
MessageAttributeNames	Falso
ReceiveRequestAttemptId	Falso
VisibilityTimeout	Falso
WaitTimeSeconds	Falso
Región	Falso

Para obtener más información, consulte ReceiveMessage en la Referencia de la API de Amazon Simple Queue Service.

SQS-DeleteMessage 1.0

Elimina el mensaje especificado de la cola especificada.

Parámetro	Obligatorio
ReceiptHandle	True
QueueUrl	True
Región	Falso

Para obtener más información, consulte DeleteMessage en la Referencia de la API de Amazon Simple Queue Service.

SQS-PurgeQueue 1.0

Elimina todos los mensajes de la cola especificada.

Parámetro	Obligatorio
QueueUrl	True
Región	Falso

Para obtener más información, consulte PurgeQueue en la Referencia de la API de Amazon Simple Queue Service.

AppConfig-GetConfiguration 1.0

Recibe información acerca de una configuración.

Parámetro	Obligatorio
Aplicación	True
Entorno	True
Configuración	True
ClientId	True
ClientConfigurationVersion	Falso
Región	Falso

Para obtener más información, consulte GetConfiguration en la Referencia de la API de AWS AppConfig.

Kinesis-PutRecord 1.0

Escribe un único registro de datos en un flujo de datos de Amazon Kinesis.

Parámetro	Obligatorio
StreamName	True

Parámetro	Obligatorio
Datos	True
PartitionKey	True
SequenceNumberForOrdering	Falso
ExplicitHashKey	Falso
Región	Falso

Para obtener más información, consulte PutRecord en la Referencia de la API de Amazon Kinesis Data Streams.

StepFunctions-StartExecution 1.0

Comienza la ejecución de la máquina de estado.

Parámetro	Obligatorio
StateMachineArn	True
Nombre	Falso
Input	Falso
Región	Falso

Para obtener más información, consulte StartExecution en la Referencia de la API de AWS Step Functions.

StepFunctions: StartSyncExecution 1.0

Comienza la ejecución de la máquina de estado sincrónico.

Parámetro	Obligatorio
StateMachineArn	True

Parámetro	Obligatorio
Nombre	Falso
Input	Falso
Región	Falso
TraceHeader	Falso

Para obtener más información, consulte StartSyncExecution en la Referencia de la API de AWS Step Functions.

StepFunctions-StopExecution 1.0

Detiene una ejecución.

Parámetro	Obligatorio
ExecutionArn	True
Causa	Falso
Error	Falso
Región	Falso

Para obtener más información, consulte StopExecution en la Referencia de la API de AWS Step Functions.

Creación de integraciones privadas para API HTTP en API Gateway

Las integraciones privadas le permiten crear integraciones de API con recursos privados en una VPC, como equilibradores de carga de aplicaciones o aplicaciones basadas en contenedores de Amazon ECS.

Puede exponer sus recursos en una VPC para que los clientes de fuera de la VPC tengan acceso mediante integraciones privadas. Puede controlar el acceso a la API mediante cualquiera de los métodos de autorización admitidos por API Gateway.

> **Note**
>
> Para crear una integración privada, primero debe crear un enlace de la VPC. Los enlaces de VPC V2 ahora son compatibles con las API HTTP y de REST. Para obtener más información acerca de los enlaces de VPC V2, consulte the section called “Enlaces de VPC V2”.

Después de crear un enlace de VPC V2, puede configurar integraciones privadas que se conecten al equilibrador de carga de aplicación, a un equilibrador de carga de red o a recursos registrados con un servicio de AWS Cloud Map.

Consideraciones

Las siguientes consideraciones pueden afectar al uso de las integraciones privadas:

- Todos los recursos deben pertenecer a la misma Cuenta de AWS. Esto incluye el equilibrador de carga o el servicio de AWS Cloud Map, el enlace de VPC y la API HTTP.
- De forma predeterminada, el tráfico de integración privada utiliza el protocolo HTTP. Para usar HTTPS, especifique una `tlsConfig`. Para elloConsola de administración de AWS, mediante la , al crear la integración privada, elija Configuración avanzada y, a continuación, indique un nombre de servidor seguro.
- En el caso de las integraciones privadas, API Gateway incluye la parte de etapa del punto de conexión de la API en la solicitud a sus recursos de backend. Por ejemplo, una solicitud a la etapa `test` de una API incluye `test/`*`route-path`* en la solicitud a su integración privada. Para eliminar el nombre de etapa de la solicitud a los recursos de backend, utilice Asignación de parámetros para sobrescribir la ruta de solicitud en `$request.path`.

Creación de una integración privada mediante un equilibrador de carga de aplicación o un equilibrador de carga de red

Antes de crear una integración privada, debe crear un enlace de VPC V2. Para obtener más información acerca de los enlaces de VPC V2, consulte the section called “Enlaces de VPC V2”.

Para crear una integración privada con un equilibrador de carga de aplicación o un equilibrador de carga de red, cree una integración de proxy HTTP, especifique el enlace de la VPC que se va a utilizar y proporcione el ARN del agente de escucha del equilibrador de carga.

El siguiente comando create-integration permite crear una integración privada que se conecte a un equilibrador de carga mediante un enlace de la VPC.

```
aws apigatewayv2 create-integration --api-id api-id --integration-type HTTP_PROXY \
    --integration-method GET --connection-type VPC_LINK \
    --connection-id VPC-link-ID \
    --integration-uri arn:aws:elasticloadbalancing:us-east-2:123456789012:listener/app/
my-load-balancer/50dc6c495c0c9188/0467ef3c8400ae65
    --payload-format-version 1.0
```

Creación de una integración privada mediante la detección de servicios de AWS Cloud Map

Antes de crear una integración privada, debe crear un enlace de VPC V2. Para obtener más información acerca de los enlaces de la VPC, consulte the section called “Enlaces de VPC V2”.

En el caso de las integraciones con AWS Cloud Map, API Gateway utiliza `DiscoverInstances` para identificar recursos. Puede utilizar parámetros de consulta para orientar recursos específicos. Los atributos de los recursos registrados deben incluir direcciones IP y puertos. API Gateway distribuye las solicitudes entre los recursos en buen estado que se devuelven desde `DiscoverInstances`. Para obtener más información, consulte DiscoverInstances en la Referencia de la API de AWS Cloud Map.

> **Note**
>
> Si utiliza Amazon ECS para rellenar entradas en AWS Cloud Map, debe configurar su tarea de Amazon ECS para utilizar registros de SRV con Amazon ECS Service Discovery o active Amazon ECS Service Connect. Para obtener más información, consulte Servicios de interconexión en la Guía para desarrolladores de Amazon Elastic Container Service.

Para crear una integración privada con AWS Cloud Map, cree una integración de proxy HTTP, especifique el enlace de la VPC que desea utilizar y proporcione el ARN del servicio de AWS Cloud Map.

El siguiente comando create-integration permite crear una integración privada que utilice la detección de servicios de AWS Cloud Map para identificar recursos:

```
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
```

Configuración de CORS de las API de HTTP en API Gateway

El uso compartido de recursos entre orígenes (CORS) es una característica de seguridad del navegador que restringe las solicitudes HTTP que se inician desde secuencias de comandos que se ejecutan en el navegador. Si no puede acceder a la API y recibe un mensaje de error que contiene `Cross-Origin Request Blocked`, es posible que deba habilitar CORS. Para obtener más información, consulte ¿Qué es el CORS?

Por lo general, se necesita CORS para crear aplicaciones web que acceden a las API alojadas en un dominio u origen distinto. Puede habilitar CORS para permitir las solicitudes a la API desde una aplicación web alojada en un dominio distinto. Por ejemplo, si la API está alojada en `https://{api_id}.execute-api.{region}.amazonaws.com/` y desea llamar a la API desde una aplicación web alojada en `example.com`, la API debe ser compatible con CORS.

Si configura CORS para una API, API Gateway envía automáticamente una respuesta a las solicitudes OPTIONS preliminares, incluso si no hay ninguna ruta OPTIONS configurada para la API. En una solicitud de CORS, API Gateway agrega los encabezados de CORS configurados a la respuesta de una integración.

> **Note**
>
> Si configura CORS para una API, API Gateway ignora los encabezados de CORS devueltos por su integración de backend.

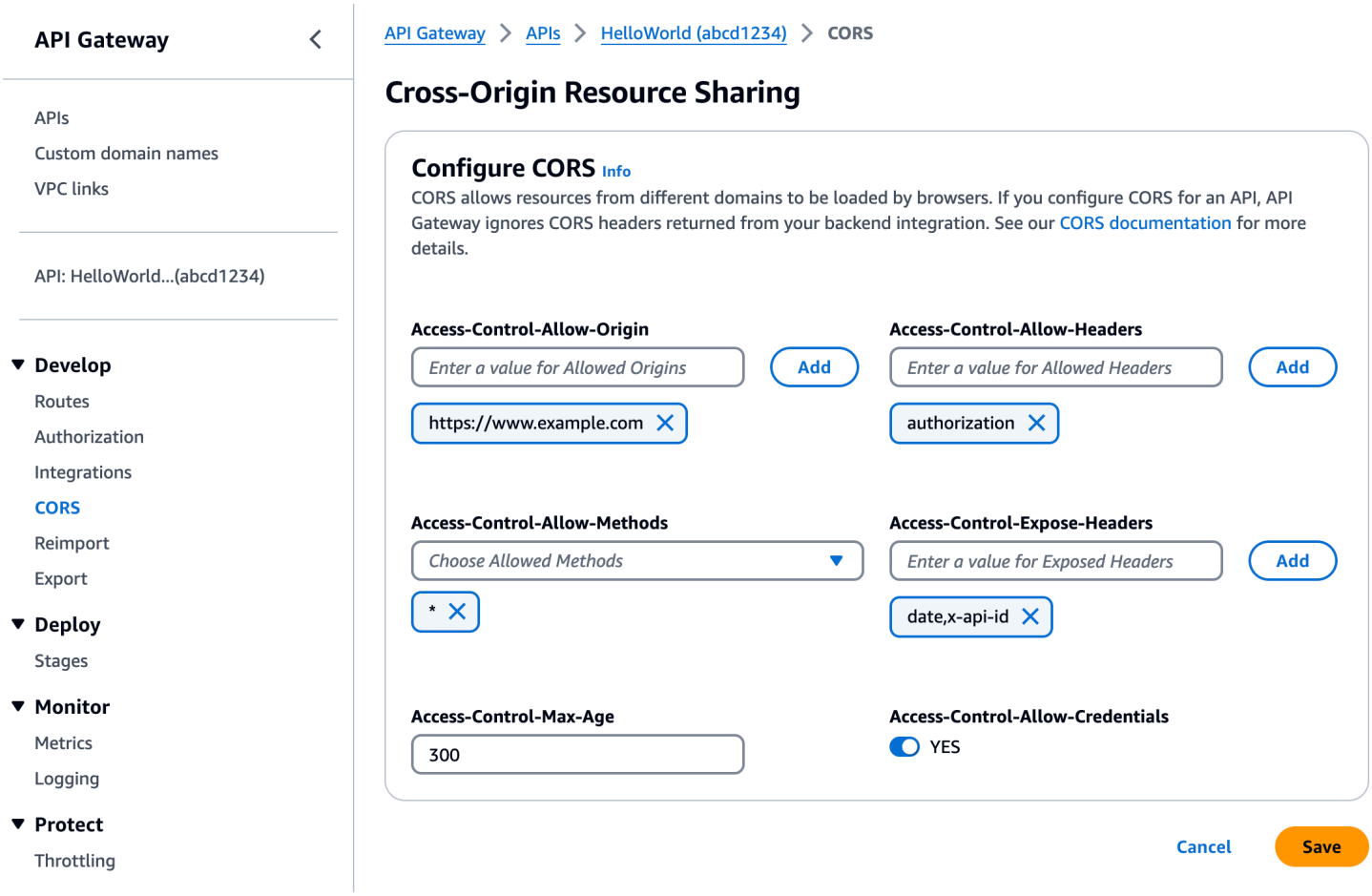
Puede especificar los siguientes parámetros en una configuración de CORS. Para agregar estos parámetros mediante la consola de API HTTP de API Gateway, seleccione Agregar después de ingresar el valor.

Encabezados de CORS	Propiedad de configuración de CORS	Valores de ejemplo
Access-Control-Allow-Origin	allowOrigins	• `https://www.example.com` • * (permitir todos los orígenes)

Encabezados de CORS	Propiedad de configuración de CORS	Valores de ejemplo
		• `https://*` (permitir cualquier origen que comience por https://) • `http://*` (permitir cualquier origen que comience por http://)
Access-Control-Allow-Credentials	allowCredentials	true
Access-Control-Expose-Headers	exposeHeaders	date, x-api-id, *
Access-Control-Max-Age	maxAge	300
Access-Control-Allow-Methods	allowMethods	GET, POST, DELETE, *
Access-Control-Allow-Headers	allowHeaders	authorization, *

Para devolver encabezados de CORS, la solicitud debe contener un encabezado `origin`. Para el método `OPTIONS`, la solicitud debe contener los encabezados `origin` y `Access-Control-Request-Method`.

Es posible que la configuración de CORS sea similar a lo siguiente:



Configuración de CORS para una API HTTP con una ruta **`$default`** y un autorizador

Puede habilitar CORS y configurar la autorización para cualquier ruta de una API HTTP. Cuando habilita CORS y la autorización para la ruta `$default`, hay que tener en cuenta algunas consideraciones especiales. La ruta `$default` captura las solicitudes de todos los métodos y rutas que no haya definido explícitamente, incluidas las solicitudes `OPTIONS`. Para admitir las solicitudes de `OPTIONS` no autorizadas, agregue una ruta de `OPTIONS /{proxy+}` a la API que no requiera autorización y asocie una integración a la ruta. La ruta `OPTIONS /{proxy+}` tiene mayor prioridad que la ruta `$default`. Como resultado, permite a los clientes enviar solicitudes `OPTIONS` a su API sin autorización. Para obtener más información acerca de las prioridades de enrutamiento, consulte Enrutamiento de solicitudes de la API.

Configuración de CORS para una API HTTP mediante la AWS CLI

El siguiente comando update-api permite habilitar las solicitudes de CORS procedentes de `https://www.example.com`:

Example

```
aws apigatewayv2 update-api --api-id api-id --cors-configuration AllowOrigins="https://
www.example.com"
```

Para obtener más información, consulte CORS en la Referencia de la API de Amazon API Gateway Versión 2.

Transformación de las solicitudes y respuestas de API para las API HTTP en API Gateway

Puede modificar las solicitudes API de los clientes antes de que lleguen a sus integraciones de backend. También puede cambiar la respuesta de las integraciones antes de que API Gateway devuelva la respuesta a los clientes. La asignación de parámetros sirve para modificar las solicitudes y respuestas de API para las API HTTP. Para utilizar la asignación de parámetros, especifique los parámetros de solicitud o respuesta de API que se van a modificar e indique cómo modificarlos.

Transformación de solicitudes de API

Los parámetros de solicitud se utilizan para cambiar las solicitudes antes de que lleguen a sus integraciones de backend. Puede modificar encabezados, cadenas de consulta o la ruta de la solicitud.

Los parámetros de solicitud son una asignación de tipo valor-clave. La clave identifica la ubicación del parámetro de solicitud que se va a cambiar y cómo cambiarlo. El valor especifica los datos nuevos para el parámetro.

En la siguiente tabla se muestran las claves admitidas.

Tipo	Sintaxis
Encabezado	`append|overwrite|remove:header.` *`headername`*
Cadena de consulta	`append|overwrite|remove:querystring.` *`querystring-name`*
Ruta	`overwrite:path`

En la siguiente tabla se muestran los valores admitidos que se pueden asignar a los parámetros.

Tipo	Sintaxis	Notas
Valor del encabezado	$request.header.*name* o ${request.header.*name*}	Los nombres de encabezado no distinguen entre mayúsculas y minúsculas. API Gateway combina varios valores de encabezado con comas, por ejemplo, `"header1" : "value1,value2"` . Algunos encabezados están reservados. Para obtener más información, consulte the section called “Encabezados reservados”.
Valor de la cadena de consulta	$request.querystring.*name* o ${request.querystring.*name*}	Los nombres de cadenas de consulta distinguen entre mayúsculas y minúsculas. API Gateway combina varios valores con comas; por ejemplo, `"querystring1" "Value1,Value2"` .
Cuerpo de la solicitud	$request.body.*name* o ${request.body.*name*}	Una expresión de ruta JSON. El descenso recursivo (`$request.body..name`) y las expresiones de filtro (`?(expression)`) no son compatibles. **Note** Cuando se especifica una ruta JSON, API Gateway trunca el cuerpo de la

Tipo	Sintaxis	Notas
		solicitud en 100 KB y, a continuación, aplica la expresión de selección. Para enviar cargas de más de 100 KB, especifiq ue `$request.body` .
Ruta de solicitud	$request.path o ${request .path}	La ruta de la solicitud, sin el nombre de la etapa.
Parámetro de ruta	$request.path.*name* o ${request.path.*name*}	El valor de un parámetro de ruta en la solicitud. Por ejemplo, si la ruta es `/pets/{petId}` , puede asignar el parámetro `petId` de la solicitud a `$request.path.petId` .
Variable de contexto	$context.*variableName* o ${context.*variableName* }	El valor de una variable de contexto. **Note** Solo se admiten los caracteres especiales `.` y `_`.
Variable de etapa	$stageVariables.*variableName* o ${stageVariables.*variableName* }	El valor de una variable de etapa.
Valor estático	*string*	Un valor constante.

> **Note**
>
> Para utilizar distintas variables en una expresión de selección, incluya la variable entre corchetes. Por ejemplo, `${request.path.name} ${request.path.id}`.

Transformación de respuestas de API

Los parámetros de respuesta se utilizan para transformar la respuesta HTTP de una integración de backend antes de devolver la respuesta a los clientes. Puede modificar los encabezados o el código de estado de una respuesta antes de que API Gateway devuelva la respuesta a los clientes.

Los parámetros de respuesta se configuran para cada código de estado que devuelve la integración. Los parámetros de respuesta son una asignación de tipo valor-clave. La clave identifica la ubicación del parámetro de solicitud que se va a cambiar y cómo cambiarlo. El valor especifica los datos nuevos para el parámetro.

En la siguiente tabla se muestran las claves admitidas.

Tipo	Sintaxis
Encabezado	`append|overwrite|remove:header.` *`headername`*
Código de estado	`overwrite:statuscode`

En la siguiente tabla se muestran los valores admitidos que se pueden asignar a los parámetros.

Tipo	Sintaxis	Notas
Valor del encabezado	$response.header.*name* o ${response.header.*name*}	Los nombres de encabezado no distinguen entre mayúsculas y minúsculas. API Gateway combina varios valores de encabezado con comas, por ejemplo, `"header1" : "value1,value2"` .

Tipo	Sintaxis	Notas
		Algunos encabezados están reservados. Para obtener más información, consulte the section called “Encabezados reservados”.
Cuerpo de respuesta	$response.body.*name* o ${response.body.*name*}	Una expresión de ruta JSON. El descenso recursivo (`$response.body..name`) y las expresiones de filtro (`?(expression)`) no son compatibles. **Note** Cuando se especifica una ruta JSON, API Gateway trunca el cuerpo de la respuesta en 100 KB y, a continuación, aplica la expresión de selección. Para enviar cargas de más de 100 KB, especifique `$response.body`.
Variable de contexto	$context.*variableName* o ${context.*variableName*}	El valor de una variable de contexto compatible.
Variable de etapa	$stageVariables.*variableName* o ${stageVariables.*variableName*}	El valor de una variable de etapa.
Valor estático	*string*	Un valor constante.

> **Note**
>
> Para utilizar distintas variables en una expresión de selección, incluya la variable entre corchetes. Por ejemplo, `${request.path.name} ${request.path.id}`.

Encabezados reservados

Los siguientes encabezados están reservados. No puede configurar asignaciones de solicitud o respuesta para estos encabezados.

- access-control-*
- apigw-*
- Autorización
- Conexión
- Content-Encoding
- Longitud del contenido
- Content-Location
- Forwarded
- Keep-Alive
- Origen
- Proxy-Authenticate
- Proxy-Authorization
- TE
- Trailers
- Transfer-Encoding
- Upgrade
- x-amz-*
- x-amzn-*
- X-Forwarded-For
- X-Forwarded-Host
- X-Forwarded-Proto
- Via

Ejemplos

En los siguientes ejemplos de la AWS CLI, se configuran asignaciones de parámetros. Para obtener plantillas de CloudFormation de ejemplo, consulte GitHub.

Adición de un encabezado a una solicitud de API

El siguiente comando create-integration permite crear un encabezado denominado `header1` a una solicitud de API antes de que llegue a su integración de backend. API Gateway rellena el encabezado con el ID de solicitud.

```
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" }'
```

Cambio de nombre de un encabezado de solicitud

El siguiente comando create-integration permite cambiar el nombre de encabezado de una solicitud de `header1` a `header2`:

```
aws apigatewayv2 create-integration \
    --api-id abcdef123 \
    --integration-type HTTP_PROXY \
    --payload-format-version 1.0 \
    --integration-uri 'https://api.example.com' \
    --integration-method ANY \
    --request-parameters '{ "append:header.header2": "$request.header.header1",
 "remove:header.header1": "''"}'
```

Cambio de la respuesta de una integración

El siguiente comando create-integration permite configurar los parámetros de respuesta de una integración. Cuando las integraciones devuelven un código de estado 500, API Gateway cambia el código de estado a 403 y agrega `header11` a la respuesta. Cuando la integración devuelve un código de estado 404, API Gateway agrega un `error` encabezado a la respuesta.

```
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"}  }'
```

Eliminación de asignaciones de parámetros configuradas

El siguiente comando update-integration permite eliminar parámetros de solicitud configurados previamente para `append:header.header1`. También elimina los parámetros de respuesta configurados previamente para un código de estado 200.

```
aws apigatewayv2 update-integration \
    --api-id abcdef123 \
    --integration-id hijk456 \
    --request-parameters '{"append:header.header1" : ""}' \
    --response-parameters '{"200" : {}}'
```

Uso de definiciones de OpenAPI para las API HTTP en API Gateway

Para definir su API HTTP, puede utilizar un archivo de definición de OpenAPI 3.0. A continuación, puede importar la definición a API Gateway para crear una API. Para obtener más información sobre las extensiones de API Gateway para OpenAPI, consulte Extensiones de OpenAPI.

Importación de una API HTTP

Para crear una API HTTP, puede importar un archivo de definición de OpenAPI 3.0.

Para migrar desde una API de REST a una API HTTP, puede exportar la API de REST como un archivo de definición de OpenAPI 3.0. A continuación, importe la definición de la API como una API HTTP. Para obtener más información acerca de cómo exportar una API de REST, consulte Exportación de una API REST desde API Gateway.

> **Note**
>
> Las API HTTP admiten las mismas variables de AWS que las API REST. Para obtener más información, consulte AWSVariables de para la importación de OpenAPI.

Importar información de validación

Al importar una API, API Gateway proporciona tres categorías de información de validación.

Información

Una propiedad es válida de acuerdo con la especificación de OpenAPI, pero dicha propiedad no es compatible con las API HTTP.

Por ejemplo, el siguiente fragmento de OpenAPI 3.0 genera información en la importación porque las API HTTP no son compatibles con la validación de solicitudes. API Gateway ignora los campos `requestBody` y `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"
        }
```

```
      }
    }
    ...
  }
  ...
}
```

Advertencia

Una propiedad o estructura no es válida de acuerdo con la especificación de OpenAPI, pero no bloquea la creación de API. Puede especificar si API Gateway debe hacer caso omiso de estas advertencias y seguir creando la API o dejar de crear la API según las advertencias.

El siguiente documento de OpenAPI 3.0 genera advertencias en la importación porque las API HTTP solo son compatibles con integraciones de proxy de Lambda y proxy de HTTP.

```
"x-amazon-apigateway-integration": {
  "type": "AWS",
  "httpMethod": "POST",
  "uri": "arn:aws:lambda:us-east-2:123456789012:function:HelloWorld",
  "payloadFormatVersion": "1.0"
}
```

Error

La especificación OpenAPI no es válida o es incorrecta. API Gateway no puede crear ningún recurso a partir del documento con formato incorrecto. Debe corregir los errores y, a continuación, intentarlo de nuevo.

La siguiente definición de API genera errores en la importación porque las API HTTP solo son compatibles con la especificación de OpenAPI 3.0.

```
{
  "swagger": "2.0.0",
  "info": {
    "title": "My API",
    "description": "An Example OpenAPI definition for Errors/Warnings/ImportInfo",
    "version": "1.0"
  }
  ...
}
```

Como otro ejemplo, mientras que OpenAPI permite a los usuarios definir una API con varios requisitos de seguridad asociados a una operación en particular, API Gateway no admite esto. Cada operación solo puede tener una autorización de IAM, un autorizador de Lambda o un autorizador JWT. Cuando se intenta modelar varios requisitos de seguridad, se produce un error.

Importación de una API con la AWS CLI

El siguiente comando import-api permite importar el archivo de definición de OpenAPI 3.0 `api-definition.json` como una API HTTP.

Example

```
aws apigatewayv2 import-api --body file://api-definition.json
```

Example

Puede importar el siguiente ejemplo de definición de OpenAPI 3.0 para crear una API HTTP.

```
{
  "openapi": "3.0.1",
  "info": {
    "title": "Example Pet Store",
    "description": "A Pet Store API.",
    "version": "1.0"
  },
  "paths": {
    "/pets": {
      "get": {
        "operationId": "GET HTTP",
        "parameters": [
          {
            "name": "type",
            "in": "query",
            "schema": {
              "type": "string"
            }
          },
          {
            "name": "page",
            "in": "query",
            "schema": {
              "type": "string"
```

```
            }
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "headers": {
              "Access-Control-Allow-Origin": {
                "schema": {
                  "type": "string"
                }
              }
            },
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Pets"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "HTTP_PROXY",
          "httpMethod": "GET",
          "uri": "http://petstore.execute-api.us-west-1.amazonaws.com/petstore/pets",
          "payloadFormatVersion": 1.0
        }
      },
      "post": {
        "operationId": "Create Pet",
        "requestBody": {
          "content": {
            "application/json": {
              "schema": {
                "$ref": "#/components/schemas/NewPet"
              }
            }
          },
          "required": true
        },
        "responses": {
          "200": {
            "description": "200 response",
```

```
            "headers": {
              "Access-Control-Allow-Origin": {
                "schema": {
                  "type": "string"
                }
              }
            },
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/NewPetResponse"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "HTTP_PROXY",
          "httpMethod": "POST",
          "uri": "http://petstore.execute-api.us-west-1.amazonaws.com/petstore/pets",
          "payloadFormatVersion": 1.0
        }
      }
    },
    "/pets/{petId}": {
      "get": {
        "operationId": "Get Pet",
        "parameters": [
          {
            "name": "petId",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "headers": {
              "Access-Control-Allow-Origin": {
                "schema": {
                  "type": "string"
```

```
                }
              }
            },
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Pet"
                }
              }
            }
          }
        },
        "x-amazon-apigateway-integration": {
          "type": "HTTP_PROXY",
          "httpMethod": "GET",
          "uri": "http://petstore.execute-api.us-west-1.amazonaws.com/petstore/pets/
{petId}",
          "payloadFormatVersion": 1.0
        }
      }
    }
  },
  "x-amazon-apigateway-cors": {
    "allowOrigins": [
      "*"
    ],
    "allowMethods": [
      "GET",
      "OPTIONS",
      "POST"
    ],
    "allowHeaders": [
      "x-amzm-header",
      "x-apigateway-header",
      "x-api-key",
      "authorization",
      "x-amz-date",
      "content-type"
    ]
  },
  "components": {
    "schemas": {
      "Pets": {
        "type": "array",
```

```
        "items": {
          "$ref": "#/components/schemas/Pet"
        }
      },
      "Empty": {
        "type": "object"
      },
      "NewPetResponse": {
        "type": "object",
        "properties": {
          "pet": {
            "$ref": "#/components/schemas/Pet"
          },
          "message": {
            "type": "string"
          }
        }
      },
      "Pet": {
        "type": "object",
        "properties": {
          "id": {
            "type": "string"
          },
          "type": {
            "type": "string"
          },
          "price": {
            "type": "number"
          }
        }
      },
      "NewPet": {
        "type": "object",
        "properties": {
          "type": {
            "$ref": "#/components/schemas/PetType"
          },
          "price": {
            "type": "number"
          }
        }
      },
      "PetType": {
```

```
        "type": "string",
        "enum": [
          "dog",
          "cat",
          "fish",
          "bird",
          "gecko"
        ]
      }
    }
  }
}
```

Exportación de API HTTP desde API Gateway

Después de crear una API HTTP, puede exportar una definición de OpenAPI 3.0 de su API desde API Gateway. Puede elegir una etapa que exportar o exportar la configuración más reciente de su API. También puede importar una definición de API exportada a API Gateway para crear otra API idéntica. Para obtener más información sobre la importación de definiciones de API, consulte Importación de una API HTTP.

Exportación de la definición de OpenAPI 3.0 de una etapa mediante AWS CLI

El siguiente comando export-api permite exportar una definición de OpenAPI de una etapa de API denominada `prod` a un archivo YAML denominado `stage-definition.yaml`. El archivo de definición exportado incluye extensiones de API Gateway de forma predeterminada.

```
aws apigatewayv2 export-api \
    --api-id api-id  \
    --output-type YAML  \
    --specification OAS30 \
    --stage-name prod \
    stage-definition.yaml
```

Exportación de una definición de OpenAPI 3.0 de los últimos cambios de la API mediante AWS CLI

El siguiente comando export-api permite exportar la definición de OpenAPI de una API HTTP a un archivo JSON denominado `latest-api-definition.json`. Dado que el comando no especifica una etapa, API Gateway exporta la configuración más reciente de la API, con independencia de si se ha implementado o no en una etapa. El archivo de definición exportado no incluye extensiones de API Gateway.

```
aws apigatewayv2 export-api \
    --api-id api-id  \
    --output-type JSON  \
    --specification OAS30 \
    --no-include-extensions \
    latest-api-definition.json
```

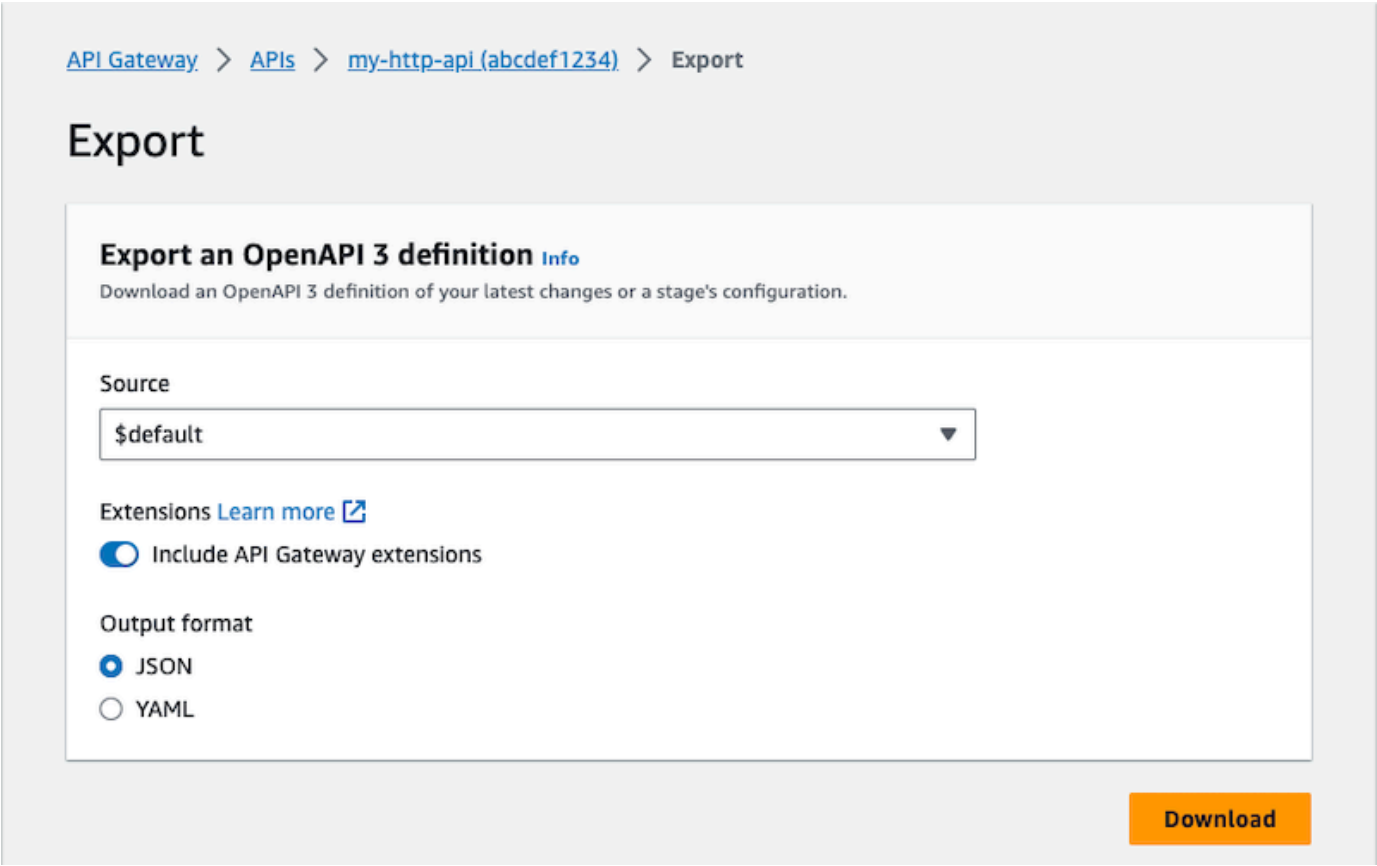
Para obtener más información, consulte ExportAPI en la Referencia de la API de Amazon API Gateway Versión 2.

Exportación de una definición de OpenAPI 3.0 mediante la consola de API Gateway

El procedimiento siguiente describe cómo exportar una definición de OpenAPI de una API HTTP.

Para exportar una definición de OpenAPI 3.0 con la consola de API Gateway

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Elija una API HTTP.
3. En el panel de navegación principal, en Desarrollar, elija Exportar.
4. Seleccione una de las siguientes opciones para exportar la API:



a. En Origen, seleccione un origen para la definición de OpenAPI 3.0. Puede elegir una etapa que exportar o exportar la configuración más reciente de la API.

b. Active Incluir extensiones de API Gateway para incluir las extensiones de API Gateway.

c. En Formato de salida, seleccione un formato de salida.

5. Elija Descargar.

Publicación de las API de HTTP para que los clientes la invoquen

Puede utilizar etapas y nombres de dominio personalizados para publicar su API para que los clientes la invoquen.

Una referencia lógica a un estado del ciclo de vida de la API (por ejemplo, `dev`, `prod`, `beta` o `v2`). Cada etapa es una referencia con nombre a una implementación de la API y está disponible para que las aplicaciones cliente la invoquen. Puede configurar diferentes integraciones y configuraciones para cada etapa de una API.

Puede usar nombres de dominio personalizados para proporcionar una URL más simple e intuitiva para que los clientes invoquen su API en vez de la URL predeterminad, `https://`*`api-id`*`.execute-api.`*`region`*`.amazonaws.com/`*`stage`*.

> **Note**
>
> Para aumentar la seguridad de las API de API Gateway, el dominio `execute-api.{`*`region`*`}.amazonaws.com` se registra en la lista de sufijos públicos (PSL). Para mayor seguridad, recomendamos que utilice cookies con un prefijo `__Host-` en caso de que necesite configurar cookies confidenciales en el nombre de dominio predeterminado de las API de API Gateway. Esta práctica le ayudará a proteger su dominio de los intentos de falsificación de solicitudes entre sitios (CSRF). Para obtener más información, consulte la página de configuración de cookies en la red de desarrolladores de Mozilla.

Temas

- Etapas de las API HTTP en API Gateway
- Política de seguridad de las API de HTTP en API Gateway
- Nombres de dominio personalizados para las API de HTTP en API Gateway

Etapas de las API HTTP en API Gateway

Una etapa de API es una referencia lógica a un estado del ciclo de vida de la API (por ejemplo, `dev`, `prod`, `beta` o `v2`). Las etapas de API se identifican por su ID de API y su nombre de etapa, y se incluyen en la URL que utiliza para invocar la API. Cada etapa es una referencia con nombre a una implementación de la API y está disponible para que las aplicaciones cliente la invoquen.

Puede crear una etapa `$default` que se ofrezca desde la base de la URL de su API, por ejemplo, `https://{api_id}.execute-api.{region}.amazonaws.com/`. Utilice esta URL para invocar una etapa de la API.

Una implementación es una instantánea de la configuración de la API. Después de implementar una API en una etapa, esta está disponible para que los clientes la invoquen. Debe implementar una API para que los cambios surtan efecto. Si habilita implementaciones automáticas, los cambios realizados en una API se liberarán automáticamente.

Uso de variables de etapa para las API de HTTP en API Gateway

Las variables de etapa son pares clave-valor que se pueden definir para una etapa de una API HTTP. Actúan como variables de entorno y se pueden usar en la configuración de la API.

Las variables de etapa no están previstas para su uso con datos confidenciales (por ejemplo, credenciales). Para transferir información confidencial a las integraciones, utilice un autorizador de AWS Lambda. Puede pasar datos confidenciales a integraciones en la salida del autorizador de Lambda. Para obtener más información, consulte the section called “Formato de respuesta del autorizador de Lambda”.

Ejemplo: Uso de una variable de etapa para personalizar el punto de conexión de integración HTTP

Por ejemplo, puede definir una variable de etapa y, a continuación, establecer su valor como punto de conexión HTTP para una integración de proxy HTTP. Posteriormente, puede hacer referencia al punto de conexión mediante el nombre de variable de etapa asociada. Al hacer esto, puede usar la misma configuración de API con un punto de conexión diferente en cada etapa. Del mismo modo, puede utilizar variables de etapa para especificar una integración de funciones de AWS Lambda diferente para cada etapa de la API.

Para utilizar una variable de etapa para personalizar el punto de conexión de integración HTTP, primero debe establecer el nombre y el valor de la variable de etapa (por ejemplo, `url`) con un valor de `example.com`. A continuación, configure una integración de proxy HTTP. En lugar de escribir la

URL del punto de conexión, puede indicar a API Gateway que use el valor de la variable de etapa, **`http://${stageVariables.url}`**. Este valor indica a API Gateway que sustituya la variable de etapa `${}` en el tiempo de ejecución en función de la etapa de la API.

Puede hacer referencia a variables de etapa de una manera similar para especificar el nombre de una función de Lambda o el ARN de un rol de AWS.

A la hora de especificar el nombre de una función de Lambda como un valor de variable de etapa, debe configurar manualmente los permisos en la función de Lambda. El siguiente comando add-permission permite configurar el permiso de la función de Lambda:

```
aws lambda add-permission --function-name arn:aws:lambda:XXXXXX:your-lambda-function-
name --source-arn arn:aws:execute-api:us-east-1:YOUR_ACCOUNT_ID:api_id/*/HTTP_METHOD/
resource --principal apigateway.amazonaws.com --statement-id apigateway-access --action
 lambda:InvokeFunction
```

Referencia de variables de etapa de API Gateway para las API HTTP en API Gateway

Puede utilizar variables de etapa de API Gateway para las API de HTTP en los siguientes casos.

URI de integración HTTP

Puede utilizar una variable de etapa como parte de una URI de integración HTTP, tal y como se muestra en los siguientes ejemplos.

- Una URI completa sin protocol – `http://${stageVariables.<variable_name>}`
- Un dominio complet – `http://${stageVariables.<variable_name>}/resource/operation`
- Un subdomini – `http://${stageVariables.<variable_name>}.example.com/resource/operation`
- Una rut – `http://example.com/${stageVariables.<variable_name>}/bar`
- Una cadena de consult – `http://example.com/foo?q=${stageVariables.<variable_name>}`

Funciones de Lambda

Puede utilizar una variable de etapa en lugar de un nombre o alias de integración de una función de Lambda, como se muestra en los ejemplos siguientes.

- `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**
>
> Para utilizar una variable de etapa para una función de Lambda, la función debe estar en la misma cuenta que la API. Las variables de etapa no admiten funciones de Lambda entre cuentas.

AWSCredenciales de integración de

Puede utilizar una variable de etapa como parte de un ARN de credenciales de usuario o rol de AWS, como se muestra en el siguiente ejemplo.

- `arn:aws:iam::<account_id>:${stageVariables.<variable_name>}`

Política de seguridad de las API de HTTP en API Gateway

API Gateway aplica una política de seguridad de `TLS_1_2` para todos los puntos de conexión de la API HTTP.

Una política de seguridad es una combinación predefinida de versión mínima de TLS y conjuntos de cifrado que ofrece Amazon API Gateway. El protocolo TLS soluciona los problemas de seguridad de red como, por ejemplo, la manipulación y el acceso no autorizado entre un cliente y el servidor. Cuando sus clientes establecen un protocolo de conexión a TLS con la API a través del dominio personalizado, la política de seguridad aplica la versión de TLS y opciones del conjunto de cifrado que sus clientes pueden elegir utilizar. Esta política de seguridad acepta el tráfico de TLS 1.2 y TLS 1.3 y rechaza el tráfico de TLS 1.0.

Protocolos y cifrados TLS compatibles para las API HTTP

En la siguiente tabla se describen los protocolos TLS compatibles con las API de HTTP.

Protocolos TLS	Política de seguridad de TLS_1_2
TLSv1.3	Sí
TLSv1.2	Sí

En la siguiente tabla se describen los cifrados de TLS que están disponibles para la política de seguridad de TLS 1_2 para las API de HTTP.

Cifrados TLS	Política de seguridad de TLS_1_2
TLS-AES-128-GCM-SHA256	Sí
TLS-AES-256-GCM-SHA384	Sí
TLS-CHACHA20-POLY1305-SHA256	Sí
ECDHE-ECDSA-AES128-GCM-SHA256	Sí
ECDHE-RSA-AES128-GCM-SHA256	Sí
ECDHE-ECDSA-AES128-SHA256	Sí
ECDHE-RSA-AES128-SHA256	Sí
ECDHE-ECDSA-AES256-GCM-SHA384	Sí
ECDHE-RSA-AES256-GCM-SHA384	Sí
ECDHE-ECDSA-AES256-SHA384	Sí
ECDHE-RSA-AES256-SHA384	Sí
AES128-GCM-SHA256	Sí

Cifrados TLS	Política de seguridad de TLS_1_2
AES128-SHA256	Sí
AES256-GCM-SHA384	Sí
AES256-SHA256	Sí

Nombre del cifrado OpenSSL y RFC

OpenSSL e IETF RFC 5246 utilizan nombres distintos para los mismos cifrados. Para obtener una lista de los nombres de los cifrados, consulte the section called “Nombre del cifrado OpenSSL y RFC”.

Información acerca de las API de REST y las API de WebSocket

Para obtener más información sobre las API de REST y las API de WebSocket, consulte the section called “Elección de una política de seguridad” y the section called “Política de seguridad de las API de WebSocket en API Gateway”.

Nombres de dominio personalizados para las API de HTTP en API Gateway

Losnombres de dominio personalizados son direcciones URL más sencillas e intuitivas que puede proporcionar a los usuarios de la API.

Después de implementar la API, usted (y sus clientes) puede invocar la API utilizando la URL base predeterminada con el siguiente formato:

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

donde *`api-id`* lo genera API Gateway, *`region`* es la región de AWS y *`stage`* lo especifica usted al implementar la API.

La parte del nombre de host de la URL, *`api-id`*`.execute-api.`*`region`*`.amazonaws.com`, hace referencia a un punto de conexión de la API. El punto de conexión de la API predeterminado se genera aleatoriamente y no es muy descriptivo.

Con los nombres de dominio personalizados, puede configurar el nombre de host de la API y elegir una ruta base (por ejemplo, `myservice`) para asignarle una URL alternativa. Por ejemplo, una URL base de la API más simple puede ser:

```
https://api.example.com/myservice
```

Consideraciones

Las siguientes consideraciones pueden afectar el uso de un nombre de dominio personalizado.

- Un nombre de dominio personalizado regional se puede asociar con las API de REST y las API de HTTP. Puede utilizar las API de API Gateway versión 2 para crear y administrar nombres de dominio de región personalizados para las API de REST.
- Para la versión de TLS mínima, solo se admite TLS 1.2.
- Debe crear o actualizar el registro de recursos del proveedor de DNS para asignarlo al punto de conexión de la API. Si no se realiza este mapeo, las solicitudes de API vinculadas al nombre de dominio personalizado no pueden llegar a API Gateway.
- Puede admitir un número casi infinito de nombres de dominio sin exceder el límite predeterminado mediante el uso de un certificado comodín. Para obtener más información, consulte the section called “Nombres de dominio personalizados comodín”.

Requisitos previos

Los siguientes son requisitos previos para crear un nombre de dominio personalizado.

Registrar un nombre de dominio

Debe disponer de un nombre de dominio de Internet registrado para poder crear nombres de dominio personalizados para sus API. Puede registrar su nombre de dominio de Internet con Amazon Route 53 o un registrador de dominios de terceros de su elección. Su nombre de dominio personalizado puede ser el nombre de un subdominio o el dominio raíz (lo que recibe el nombre de “vértice de zona”) de un dominio de Internet registrado.

El nombre de dominio debe seguir la especificación RFC 1035 y puede tener un máximo de 63 octetos por etiqueta y 255 octetos en total.

Certificados para nombres de dominio personalizados

Antes de configurar un nombre de dominio personalizado para una API, debe disponer de un certificado SSL/TLS listo en ACM. Si ACM no está disponible en la región de AWS donde va a crear su nombre de dominio personalizado, debe importar un certificado en API Gateway en esa región.

Para importar un certificado SSL/TLS, debe proporcionar el cuerpo del certificado SSL/TLS con formato PEM, su clave privada y la cadena de certificados para el nombre de dominio personalizado.

Los certificados almacenados en ACM se identifican mediante su ARN. Con los certificados emitidos por ACM no tiene que preocuparse de si se exponen datos del certificado confidenciales, como la clave privada. Para utilizar un certificado administrado por AWS para un nombre de dominio, solo tiene que hacer referencia a su ARN.

Si la aplicación utiliza asignación de certificados, también conocido como asignación SSL, para asignar un certificado de ACM, es posible que la aplicación no se pueda conectar al dominio una vez que AWS renueva el certificado. Para obtener más información, consulte Problemas de asignación de certificados en la Guía del usuario de AWS Gestor de certificados.

Nombres de dominio personalizados comodín

Con los nombres de dominio personalizados comodín, puede admitir un número casi infinito de nombres de dominio sin exceder el límite predeterminado. Por ejemplo, puede dar a cada uno de los clientes su propio nombre de dominio, *`customername`*`.api.example.com`.

Para crear un nombre de dominio personalizado comodín, especifique un comodín (`*`) como primer subdominio de un dominio personalizado que represente todos los subdominios posibles de un dominio raíz.

Por ejemplo, el nombre de dominio personalizado comodín `*.example.com` produce subdominios como `a.example.com`, `b.example.com` y `c.example.com`, que enrutan al mismo dominio.

Los nombres de dominio personalizados comodín admiten configuraciones distintas de los nombres de dominio personalizados estándar de API Gateway. Por ejemplo, en una sola cuenta de AWS, puede configurar `*.example.com` y `a.example.com` para que se comporten de manera diferente.

Para crear un nombre de dominio personalizado comodín, debe proporcionar un certificado emitido por ACM que se haya validado utilizando el método DNS o de validación de correo electrónico.

> **Note**
>
> No se puede crear un nombre de dominio personalizado comodín si una cuenta de AWS diferente ha creado un nombre de dominio personalizado que entra en conflicto con el nombre de dominio personalizado comodín. Por ejemplo, si la cuenta A ha creado `a.example.com`, la cuenta B no puede crear el nombre de dominio personalizado comodín `*.example.com`.
> Si la cuenta A y la cuenta B comparten un propietario, puede contactar con el Centro de soporte de AWS para solicitar una excepción.

Siguientes pasos para nombres de dominio personalizados

Para configurar un nombre de dominio personalizado para una API de HTTP, utilice la sección API de REST de la Guía para desarrolladores de API Gateway.

En primer lugar, especifique el certificado para el nombre de dominio personalizado. Para obtener más información, consulte the section called “Preparación de certificados en AWS Gestor de certificados”. A continuación, cree un nombre de dominio personalizado regional. Para obtener más información, consulte the section called “Configuración de un nombre de dominio personalizado regional”.

Asignación de etapas de API a un nombre de dominio personalizado para las API HTTP

Las asignaciones de API se utilizan para conectar etapas de la API a un nombre de dominio personalizado. Después de crear un nombre de dominio y configurar registros DNS, se utilizan asignaciones de API para enviar tráfico a las API a través del nombre de dominio personalizado.

Una asignación de API especifica una API, una etapa y, de forma opcional, una ruta que se utilizará para la asignación. Por ejemplo, puede asignar la etapa de `production` de una API a `https://api.example.com/orders`.

Puede asignar etapas de API HTTP y de API de REST al mismo nombre de dominio personalizado.

Antes de crear una asignación de API, debe tener una API, una etapa y un nombre de dominio personalizado. Para obtener más información sobre cómo crear un nombre de dominio personalizado, consulte the section called “Configuración de un nombre de dominio personalizado regional”.

Enrutamiento de solicitudes de la API

Puede configurar asignaciones de API con varios niveles; por ejemplo `orders/v1/items` y `orders/v2/items`.

En el caso de las asignaciones de API con varios niveles, API Gateway enruta las solicitudes a la asignación de API con la ruta coincidente más larga. API Gateway solo considera las rutas configuradas para las asignaciones de API, y no las rutas de la API, para seleccionar la API que se va a invocar. Si ninguna ruta coincide con la solicitud, API Gateway envía la solicitud a la API que ha asignado a la ruta vacía (`none`).

En el caso de los nombres de dominio personalizados que utilizan asignaciones de API con varios niveles, API Gateway enruta las solicitudes a la asignación de API con el prefijo coincidente más largo.

Por ejemplo, pensemos en un nombre de dominio personalizado `https://api.example.com` con las siguientes asignaciones de API:

1. (`none`) asignado a la API 1.
2. `orders` asignado a la API 2.
3. `orders/v1/items` asignado a la API 3.
4. `orders/v2/items` asignado a la API 4.
5. `orders/v2/items/categories` asignado a la API 5.

Solicitud	API seleccionada	Explicación
`https://api.example.com/orders`	`API 2`	La solicitud coincide exactamente con esta asignación de API.
`https://api.example.com/orders/v1/items`	`API 3`	La solicitud coincide exactamente con esta asignación de API.
`https://api.example.com/orders/v2/items`	`API 4`	La solicitud coincide exactamente con esta asignación de API.

Solicitud	API seleccionada	Explicación
`https://api.example.com/orders/v1/items/123`	`API 3`	API Gateway elige la asignación con la ruta de coincidencia más larga. El `123` al final de la solicitud no afecta a la selección.
`https://api.example.com/orders/v2/items/categories/5`	`API 5`	API Gateway elige la asignación con la ruta de coincidencia más larga.
`https://api.example.com/customers`	`API 1`	API Gateway utiliza la asignación vacía como un catch-all.
`https://api.example.com/ordersandmore`	`API 2`	API Gateway elige la asignación con el prefijo de coincidencia más largo. En el caso de un nombre de dominio personalizado configurado con asignaciones de un solo nivel, por ejemplo, solo `https://api.example.com/orders` y `https://api.example.com/`, API Gateway elegiría `API 1`, ya que no hay una ruta que coincida con `ordersandmore`.

Restricciones

- En el caso de una asignación de API, el nombre de dominio personalizado y las API asignadas deben pertenecer a la misma cuenta de AWS.

- Las asignaciones de API deben contener solo letras, números y los siguientes caracteres: `$-_.+!*'()/`.
- La longitud máxima de la ruta en una asignación de API es de 300 caracteres.
- Puede tener 200 asignaciones de API con varios niveles para cada nombre de dominio. Este límite no incluye la asignación de la API con niveles únicos, como `/prod`.
- Solo puede asignar API HTTP a un nombre de dominio regional personalizado con la política de seguridad de TLS 1.2.
- No puede asignar las API de WebSocket al mismo nombre de dominio personalizado que una API HTTP o API REST.
- Si crea una asignación de API con varios niveles, API Gateway convierte todos los nombres de encabezado a minúsculas.

Creación de una asignación de API

Para crear una asignación de API, primero debe crear un nombre de dominio personalizado, una API y una etapa. Para obtener información sobre cómo crear un nombre de dominio personalizado, consulte the section called “Configuración de un nombre de dominio personalizado regional”.

Para ver ejemplos de las plantillas de AWS Serverless Application Model que crean todos los recursos, consulte Sessions With SAM en GitHub.

Consola de administración de AWS

Para crear una asignación de API

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Elija Nombres de dominio personalizados.
3. Seleccione un nombre de dominio personalizado que ya haya creado.
4. Elija Asignaciones de API.
5. Elija Configurar asignaciones de API.
6. Seleccione Agregar nueva asignación.
7. Introduzca una API, una Etapa y, de forma opcional, una Ruta.
8. Seleccione Save.

AWS CLI

El siguiente comando create-api-mapping permite crear una asignación de API. En este ejemplo, API Gateway envía solicitudes a `api.example.com/v1/orders` a la API y a la etapa especificadas.

```
aws apigatewayv2 create-api-mapping \
    --domain-name api.example.com \
    --api-mapping-key v1/orders \
    --api-id a1b2c3d4 \
    --stage test
```

CloudFormation

En el siguiente ejemplo de CloudFormation, se crea una asignación de API.

```
MyApiMapping:
  Type: 'AWS::ApiGatewayV2::ApiMapping'
  Properties:
    DomainName: api.example.com
    ApiMappingKey: 'orders/v2/items'
    ApiId: !Ref MyApi
    Stage: !Ref MyStage
```

Deshabilitación del punto de conexión predeterminado para las API HTTP

De forma predeterminada, los clientes pueden invocar su API mediante el punto de conexión `execute-api` que API Gateway genera para ella. Para asegurarse de que los clientes solo puedan acceder a su API mediante un nombre de dominio personalizado, deshabilite el punto de conexión predeterminado `execute-api`. Cuando deshabilita el punto de conexión predeterminado, esto afecta a todas las etapas de una API.

El siguiente procedimiento muestra cómo deshabilitar el punto de conexión predeterminado para una API HTTP.

Consola de administración de AWS

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Elija una API HTTP.

3. Elija el ID de la API para abrir la página Detalles de la API.
4. En Detalles de la API, elija Editar.
5. En Punto de conexión predeterminado, elija Deshabilitar.
6. Seleccione Guardar.

   Si activa las implementaciones automáticas para la etapa, no necesita volver a implementar la API con el fin de que el cambio surta efecto. De lo contrario, tendrá que volver a implementar la API.
7. (Opcional) Elija Implementar y, a continuación, vuelva a implementar la API o cree una nueva etapa para que el cambio surta efecto.

AWS CLI

El siguiente comando update-domain-name deshabilita el punto de conexión predeterminado de una API HTTP.

```
aws apigatewayv2 update-api \
    --api-id abcdef123 \
    --disable-execute-api-endpoint
```

Tras deshabilitar el punto de conexión predeterminado, debe implementar la API para que el cambio surta efecto, salvo que se hayan habilitado las implementaciones automáticas.

El siguiente comando create-deployment crea una implementación:

```
aws apigatewayv2 create-deployment \
    --api-id abcdef123 \
    --stage-name dev
```

Tipos de direcciones IP para nombres de dominio personalizados para API HTTP

Al crear una API, se especifica el tipo de direcciones IP que puede invocar el dominio. Puede elegir IPv4 para resolver las direcciones IPv4 para invocar el dominio o puede elegir pila doble para permitir que las direcciones IPv4 e IPv6 invoquen el dominio. Le recomendamos que establezca el tipo de dirección IP en pila doble para mitigar el agotamiento del espacio IP o para mejorar la postura de seguridad. Para obtener más información sobre los beneficios de un tipo de dirección IP de pila doble, consulte IPv6 en AWS.

Consideraciones para los tipos de direcciones IP

Es posible que las siguientes consideraciones afecten al uso de los tipos de direcciones IP.

- El tipo de dirección IP predeterminado para los nombres de dominio personalizados de API Gateway es IPv4.
- El nombre de dominio personalizado no necesita tener el mismo tipo de dirección IP para todas las API asignadas a él. Si desactiva el punto de conexión de la API predeterminado, esto podría afectar a la forma en que los intermediarios pueden invocar a la API.

Cambio del tipo de dirección IP de un nombre de dominio personalizado

Puede cambiar el tipo de dirección IP actualizando la configuración de punto de conexión del dominio. Puede actualizar la configuración de punto de conexión del dominio mediante el uso de la Consola de administración de AWS, la AWS CLI, CloudFormation, o un AWS SDK.

Consola de administración de AWS

Cambio del tipo de dirección IP de un nombre de dominio personalizado

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Elija un nombre de dominio personalizado público.
3. Elija la Configuración de punto de conexión.
4. Para tipo de dirección IP, seleccione IPv4 o Pila doble.
5. Seleccione Save.

AWS CLI

El siguiente comando update-domain-name actualiza una API para que tenga un tipo de dirección IP de pila doble:

```
aws apigatewayv2 update-domain-name \
   --domain-name dualstack.example.com \
   --domain-name-configurations CertificateArn=arn:aws:acm:us-
east-1:111122223333:certificate/abcd1234-5678-abc,IpAddressType=dualstack
```

El resultado será similar al siguiente:

```
{
```

```
    "ApiMappingSelectionExpression": "$request.basepath",
    "DomainName": "dualstack.example.com",
    "DomainNameConfigurations": [
        {
            "ApiGatewayDomainName": "d-abcd1234.execute-api.us-
east-1.amazonaws.com",
            "CertificateArn": "arn:aws:acm:us-east-1:111122223333:certificate/
abcd1234-5678-abc",
            "DomainNameStatus": "AVAILABLE",
            "EndpointType": "REGIONAL",
            "HostedZoneId": "Z3LQWSYCGH4ADY",
            "SecurityPolicy": "TLS_1_2",
            "IpAddressType": "dualstack"
        }
    ],
    "Tags": {}
}
```

Protección de sus API de HTTP en API Gateway

API Gateway proporciona una serie de formas de proteger su API de ciertas amenazas, como usuarios malintencionados o picos de tráfico. Puede proteger su API mediante estrategias como establecer objetivos de limitación y habilitar TLS mutuo. En esta sección puede aprender cómo habilitar estas capacidades mediante API Gateway.

Temas

- Limitación de las solicitudes a sus API HTTP para mejorar el rendimiento en API Gateway
- Cómo habilitar la autenticación TLS mutua para sus API de HTTP en API Gateway

Limitación de las solicitudes a sus API HTTP para mejorar el rendimiento en API Gateway

Puede configurar la limitación de las API para evitar que se vean desbordadas por un número excesivo de solicitudes. La limitación se aplica como medida compensatoria y se debe considerar como un valor objetivo, más que como un límite de solicitudes garantizado.

API Gateway limita las solicitudes a la API utilizando el algoritmo de bucket de tokens, donde un token cuenta por una solicitud. En concreto, API Gateway examina el ratio de solicitudes y una

ráfaga de envíos de solicitudes para todas las API de su cuenta, por región. En el algoritmo de bucket de tokens, una ráfaga puede permitir que se sobrepasen los límites predefinidos, pero también hay otros factores que pueden hacer que se sobrepasen los límites en algunos casos.

Cuando los envíos de solicitudes superan los límites de ratio de solicitudes en estado estable y de ráfaga, API Gateway comienza a limitar las solicitudes. Los clientes pueden recibir respuestas de error `429 Too Many Requests` en este momento. Tras capturar estas excepciones, el cliente puede reenviar las solicitudes que han producido un error de forma que limite el ratio.

Como desarrollador de la API, puede configurar los límites objetivos para las etapas o rutas individuales de la API con el fin de mejorar el rendimiento general de todas las API de su cuenta.

Limitación de nivel de cuenta por región

De forma predeterminada, API Gateway limita las solicitudes de estado constante por segundo (RPS) en todas las API de una cuenta de AWS, por región. También limita la ráfaga (es decir, el tamaño máximo del bucket) en todas las API dentro de una cuenta de AWS, por región. En API Gateway, el límite de ráfaga representa el número máximo de envíos de solicitudes simultáneas que API Gateway; abordará antes de devolver respuestas de error `429 Too Many Requests`. Para obtener más información sobre las cuotas de limitación, consulte Cuotas y notas importantes.

Los límites por cuenta se aplican a todas las API de una cuenta en una región determinada. El límite de ratio de nivel de cuenta se puede aumentar previa solicitud. Los límites más altos son posibles con API que tienen tiempos de espera más cortos y cargas útiles más pequeñas. Para solicitar un aumento de los límites de limitación de nivel de cuenta por región, contacte con el Centro de soporte de AWS. Para obtener más información, consulte Cuotas y notas importantes. Tenga en cuenta que estos límites no pueden ser superiores a los límites de limitación de AWS.

Limitación de nivel de ruta

Puede definir la limitación de nivel de ruta para invalidar los límites de limitación de las solicitudes de nivel de cuenta de una determinada etapa o de las rutas individuales de la API. Los límites de limitación de rutas predeterminados no pueden superar los límites de ratio a nivel de cuenta.

Puede configurar la limitación de nivel de ruta mediante la AWS CLI. El siguiente comando update-stage permite configurar la limitación de nivel de ruta para la etapa y la ruta especificadas de una API.

```
aws apigatewayv2 update-stage \
```

```
    --api-id a1b2c3d4 \
    --stage-name dev \
    --route-settings '{"GET /pets":
{"ThrottlingBurstLimit":100,"ThrottlingRateLimit":2000}}'
```

Cómo habilitar la autenticación TLS mutua para sus API de HTTP en API Gateway

La autenticación TLS mutua requiere la autenticación bidireccional entre el cliente y el servidor. Con la TLS mutua, los clientes deben presentar certificados X.509 para verificar su identidad y acceder a su API. La TLS mutua es un requisito frecuente para el Internet de las cosas (IoT) y las aplicaciones entre empresas.

Puede utilizar TLS mutua junto con otras operaciones de autorización y autenticación compatibles con API Gateway. API Gateway reenvía los certificados que los clientes proporcionan a los autorizadores de Lambda y a las integraciones del backend.

> **Important**
>
> De forma predeterminada, los clientes pueden invocar su API mediante el punto de conexión `execute-api` que API Gateway genera para su API. Para asegurarse de que los clientes solo puedan acceder a su API mediante un nombre de dominio personalizado con la TLS mutua, deshabilite el punto de conexión `execute-api` predeterminado. Para obtener más información, consulte the section called “Deshabilitación del punto de conexión predeterminado”.

Requisitos previos de la TLS mutua

Para configurar la TLS mutua necesita lo siguiente:

- Un nombre de dominio personalizado
- Al menos un certificado configurado en AWS Gestor de certificados para el nombre de dominio personalizado
- Un almacén de confianza configurado y cargado en Amazon S3

Nombres de dominio personalizados

Para habilitar la TLS mutua para una API HTTP, debe configurar un nombre de dominio personalizado para la API. Puede habilitar la TLS mutua para un nombre de dominio personalizado y, a continuación, proporcionar el nombre de dominio personalizado a los clientes. Para acceder a una API mediante un nombre de dominio personalizado que tenga habilitada la TLS mutua, los clientes deben presentar certificados en los que confíe en las solicitudes de API. Dispone de más información en the section called “Nombres de dominio personalizados”.

Uso de certificados emitidos por AWS Gestor de certificados

Puede solicitar un certificado de confianza pública directamente desde ACM o importar certificados públicos o autofirmados. Para configurar un certificado en ACM, vaya a ACM. Si desea importar un certificado, siga leyendo en la siguiente sección.

Uso de un certificado importado o de AWS Autoridad de certificación privada

Para utilizar un certificado importado en ACM o un certificado de AWS Autoridad de certificación privada con TLS mutua, API Gateway necesita un `ownershipVerificationCertificate` emitido por ACM. Este certificado de propiedad solo se utiliza para comprobar que dispone de permisos para utilizar el nombre de dominio. No se utiliza en el protocolo de enlace TLS. Si todavía no tiene un `ownershipVerificationCertificate`, vaya a https://console.aws.amazon.com/acm/ para configurarlo.

Este certificado deberá mantener su validez durante toda la vida útil del nombre de dominio. Si un certificado caduca y falla la renovación automática, se bloquearán todas las actualizaciones del nombre de dominio. Tendrá que actualizar el `ownershipVerificationCertificateArn` con un `ownershipVerificationCertificate` válido para poder realizar otros cambios. El `ownershipVerificationCertificate` no se puede utilizar como certificado de servidor para otro dominio de TLS mutua en API Gateway. Si un certificado se vuelve a importar directamente en ACM, el emisor debe ser el mismo.

Configuración del almacén de confianza

Los almacenes de confianza son archivos de texto cuya extensión es `.pem`. Son una lista de certificados de confianza procedentes de entidades de certificación. Para utilizar la TLS mutua, cree un almacén de confianza de certificados X.509 en los que confíe para acceder a su API.

En el almacén de confianza debe incluir la cadena de confianza completa, desde el certificado de entidad de certificación emisora hasta el certificado de entidad de certificación raíz. API Gateway

acepta certificados de cliente emitidos por cualquier entidad de certificación presente en la cadena de confianza. Los certificados pueden ser de autoridades de certificación públicas o privadas. Los certificados pueden tener una longitud máxima de cadena de cuatro. También puede proporcionar certificados autofirmados. Se admiten los siguientes algoritmos hash en el almacén de confianza:

- SHA-256 o más seguro
- RSA-2048 o más seguro
- ECDSA-256 o más seguro

API Gateway valida una serie de propiedades de certificado. Puede utilizar autorizadores de Lambda para realizar comprobaciones adicionales cuando un cliente invoque una API, tales como verificar si se ha revocado un certificado. API Gateway valida las siguientes propiedades:

Validación	Descripción
Sintaxis X.509	El certificado debe cumplir los requisitos de sintaxis X.509.
Integridad	El contenido del certificado no debe haberse alterado con respecto al firmado por la entidad de certificación del almacén de confianza.
Validez	El período de validez del certificado debe ser actual.
Encadenamiento de nombres/encadenamiento de claves	Los nombres y asuntos de los certificados deben formar una cadena ininterrumpida. Los certificados pueden tener una longitud máxima de cadena de cuatro.

Carga del almacén de confianza a un bucket de Amazon S3 en un solo archivo

Example certificates.pem

```
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
```

```
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
...
```

El siguiente comando cp de la AWS CLI permite cargar `certificates.pem` en el bucket de Amazon S3.

```
aws s3 cp certificates.pem s3://bucket-name
```

Configuración de la TLS mutua para un nombre de dominio personalizado

Para configurar la TLS mutua para una API HTTP, debe utilizar un nombre de dominio personalizado regional para la API, y la versión mínima de TLS debe ser la 1.2. Para obtener más información sobre cómo crear y configurar un nombre de dominio personalizado, consulte the section called "Configuración de un nombre de dominio personalizado regional".

> **Note**
>
> La TLS mutua no es compatible con las API privadas.

Después de cargar su almacén de confianza en Amazon S3, puede configurar su nombre de dominio personalizado para que utilice TLS mutua. El siguiente comando create-domain-name permite crear un nombre de dominio personalizado con TLS mutua:

```
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
```

Después de crear el nombre de dominio, debe configurar los registros DNS y los mapeos de ruta base para las operaciones de la API. Para obtener más información, consulte Configuración de un nombre de dominio personalizado regional en API Gateway.

Invocar una API mediante un nombre de dominio personalizado que requiere la TLS mutua

Para invocar una API con la TLS mutua habilitada, los clientes deben presentar un certificado de confianza en la solicitud de la API. Cuando un cliente intenta invocar la API, API Gateway busca el emisor del certificado del cliente en el almacén de confianza. Para que API Gateway atienda la solicitud, en el almacén de confianza deben constar el emisor del certificado y la cadena de confianza completa hasta el certificado de entidad de certificación raíz.

El siguiente ejemplo de comando `curl` envía una solicitud a `api.example.com`, que incluye `my-cert.pem` en la solicitud. `my-key.key` es la clave privada del certificado.

```
curl -v --key ./my-key.key --cert ./my-cert.pem api.example.com
```

Su API solo se invoca si su almacén de confianza confía en el certificado. Las siguientes condiciones provocarán que API Gateway no pueda completar el protocolo de enlace TLS y deniegue la solicitud con un código de estado `403`. Si el certificado:

- no es de confianza
- ha caducado
- no utiliza un algoritmo compatible

> **Note**
>
> API Gateway no comprueba si se ha revocado un certificado.

Actualización de su almacén de confianza

Para actualizar los certificados del almacén de confianza, cargue un nuevo paquete de certificados en Amazon S3. Después, puede actualizar el nombre de dominio personalizado para que utilice el certificado actualizado.

Utilice el control de versiones de Amazon S3 para mantener varias versiones de su almacén de confianza. Cuando actualiza el nombre de dominio personalizado para utilizar una nueva versión del almacén de confianza, API Gateway devuelve advertencias si los certificados no son válidos.

API Gateway produce advertencias de certificado solo cuando actualiza el nombre de dominio. API Gateway no le notifica si caduca un certificado cargado anteriormente.

El siguiente comando update-domain-name permite actualizar un nombre de dominio personalizado para utilizar una nueva versión del almacén de confianza.

```
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'
```

Deshabilitar la TLS mutua

Para deshabilitar la TLS mutua para un nombre de dominio personalizado, elimine el almacén de confianza del nombre de dominio personalizado, como se muestra en el siguiente comando.

El siguiente comando update-domain-name permite actualizar un nombre de dominio personalizado para eliminar de él el almacén de confianza:

```
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=''
```

Solución de problemas de TLS mutuo para su API de HTTP

A continuación se proporcionan consejos para solucionar errores y problemas que puedan surgir al activar TLS mutuo.

Solución de problemas de advertencias de certificados

Cuando se crea un nombre de dominio personalizado con TLS mutua, API Gateway devuelve advertencias si los certificados del almacén de confianza no son válidos. Esto también puede ocurrir cuando se actualiza un nombre de dominio personalizado para que utilice un nuevo almacén de confianza. Las advertencias indican el problema con el certificado y el asunto del certificado que produjo la advertencia. La TLS mutua aún está habilitada en la API, pero es posible que algunos clientes no puedan acceder a su API.

Debe descodificar los certificados del almacén de confianza para identificar cuál de ellos ha producido la advertencia. Puede utilizar herramientas como `openssl` para descodificar los certificados e identificar sus asuntos.

El siguiente comando muestra el contenido de un certificado, incluido su asunto:

```
openssl x509 -in certificate.crt -text -noout
```

Actualice o elimine los certificados que produjeron advertencias y, a continuación, cargue un nuevo almacén de confianza en Amazon S3. Después de cargar el nuevo almacén de confianza, actualice el nombre de dominio personalizado para que utilice ese nuevo almacén de confianza.

Solución de problemas por conflictos de nombres de dominio

El error `"The certificate subject <certSubject> conflicts with an existing certificate from a different issuer."` significa que varias entidades de certificación han emitido un certificado para este dominio. Para cada asunto del certificado, solo puede haber un emisor en API Gateway para dominios de TLS mutua. Tendrá que obtener todos los certificados para ese asunto a través de un solo emisor. Si el problema se debe a un certificado que no está bajo su control pero puede demostrar la propiedad del nombre de dominio, contacte con Soporte para abrir un ticket.

Solución de problemas por mensajes de estado de nombres de dominio

`PENDING_CERTIFICATE_REIMPORT`: esto significa que ha vuelto a importar un certificado en ACM y ha fallado la validación porque el nuevo certificado tiene un SAN (nombre alternativo de sujeto) que no está cubierto por el `ownershipVerificationCertificate`, o bien porque el asunto o los SAN del certificado no cubren el nombre de dominio. Es posible que haya algo que esté configurado incorrectamente o que se haya importado un certificado no válido. Debe volver a importar un certificado válido en ACM. Para obtener más información sobre la validación, consulte Validación de la propiedad de dominios.

`PENDING_OWNERSHIP_VERIFICATION`: esto significa que el certificado verificado previamente ha caducado y ACM no lo ha podido renovar automáticamente. Tendrá que renovar el certificado o solicitar otro nuevo. Dispone de más información sobre la renovación de certificados en la guía Solución de problemas de renovación de certificados administrados de ACM.

Supervisión de las API de HTTP en API Gateway

Puede utilizar las métricas de CloudWatch y CloudWatch Logs para monitorear las API HTTP. Al combinar registros y métricas, puede registrar errores y monitorear el rendimiento de su API.

> **Note**
>
> API Gateway podría no generar registros y métricas en los siguientes casos:
>
> - Errores 413: Entidad de solicitud demasiado grande
> - Errores 429: Demasiadas Solicitudes
> - Errores de la serie 400 de solicitudes enviadas a un dominio personalizado que no tiene asignación de API
> - Errores de la serie 500 causados por errores internos

Temas

- Supervisión de las métricas de CloudWatch para las API de HTTP en API Gateway
- Configuración del registro de API HTTP en API Gateway

Supervisión de las métricas de CloudWatch para las API de HTTP en API Gateway

Puede monitorear la ejecución de la API mediante CloudWatch, que recopila y procesa datos sin formato de API Gateway en métricas legibles casi en tiempo real. Estas estadísticas se registran durante un periodo de 15 meses, para que pueda obtener acceso a información de historial y obtener una mejor perspectiva acerca del desempeño de su aplicación web o servicio. De forma predeterminada, los datos de las métricas de API Gateway se envían automáticamente a CloudWatch en períodos de un minuto. Para supervisar sus métricas, cree un panel de CloudWatch para la API. Para obtener más información sobre cómo crear un panel de CloudWatch, consulte Creación de un panel de CloudWatch en la Guía del usuario de Amazon CloudWatch. Para obtener más información, consulte ¿Qué es Amazon CloudWatch? en la guía del usuario de Amazon Cloudwatch.

Las siguientes métricas son compatibles con las API HTTP. También puede habilitar métricas detalladas para escribir métricas en el nivel de ruta en Amazon CloudWatch.

Métrica	Descripción
4xx	El número de errores del cliente capturados en un periodo determinado.
5xx	El número de errores del servidor capturados en un periodo determinado.
Recuento	El número total de solicitud es de API en un periodo de tiempo determinado.
IntegrationLatency	Es el tiempo entre que API Gateway transmite una solicitud al backend y se recibe una respuesta del backend.
Latency (Latencia)	El tiempo entre que API Gateway recibe una solicitud de un cliente y devuelve una respuesta al cliente. La latencia incluye la latencia de la integración y otra carga de API Gateway.
Datos procesados	La cantidad de datos procesados en bytes.

Puede usar las dimensiones de la tabla siguiente para filtrar las métricas de API Gateway.

Dimensión	Descripción
ApiId	Filtra las métricas de API Gateway para una API con el ID de API especificado.
ApiId, Stage	Filtra las métricas de API Gateway para una etapa de API cuyos ID de API y de etapa se hayan especificado.
ApiId, método, recurso, etapa	Filtra las métricas de API Gateway para un método de API con el ID de API, el ID de etapa, la ruta de recursos y el ID de enrutamiento específicos. API Gateway no enviará estas métricas a menos que haya habilitado explícitamente las métricas detalladas de CloudWatch. Puede hacerlo mediante una llamada a la acción UpdateStage de la API de REST de API Gateway V2 para actualizar la propiedad `detailedMetricsEnabled` en `true`. También puede llamar al comando update-stage AWS CLI para actualizar la propiedad `DetailedMetricsEnabled` a `true`. Si activa estas métricas, se le cobrarán cargos adicionales en su cuenta. Para obtener más

Dimensión	Descripción
	información sobre precios, consulte Precios de Amazon CloudWatch.

Configuración del registro de API HTTP en API Gateway

Puede activar el registro para escribir registros en CloudWatch Logs. Puede utilizar variables de registro para personalizar el contenido de los registros.

Para aumentar la seguridad, le recomendamos que escriba los registros en Registros de CloudWatch para todas las etapas de la API HTTP. Es probable que deba hacerlo para asegurar el cumplimiento de diversos marcos normativos. Para obtener más información, consulte Controles de Amazon API Gateway en la Guía del usuario de AWS Security Hub.

Para activar el registro para una API HTTP, tendrá que hacer lo siguiente.

1. Asegúrese de que el usuario tenga los permisos necesarios para activar el registro.
2. Cree un grupo de registros de Registros de CloudWatch.
3. Proporcione el ARN del grupo de registros de Registros de CloudWatch para una etapa de la API.

Permisos para activar el registro

Para activar el registro para una API, el usuario debe tener los siguientes permisos.

Example

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
                "logs:GetLogEvents",
```

```
                "logs:FilterLogEvents"
            ],
            "Resource": "arn:aws:logs:us-east-2:123456789012:log-group:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:PutResourcePolicy",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:CreateLogGroup",
                "logs:DescribeResourcePolicies",
                "logs:GetLogDelivery",
                "logs:ListLogDeliveries"
            ],
            "Resource": "*"
        }
    ]
}
```

Crear un grupo de registro y activar el registro para las API HTTP

Puede crear un grupo de registro y activar el registro de acceso mediante Consola de administración de AWS o AWS CLI.

Consola de administración de AWS

1. Cree un grupo de registros.

   Para obtener información acerca de cómo crear un grupo de registro mediante la consola, consulte Crear un grupo de registro en la Guía del usuario de Registros de Amazon CloudWatch.
2. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
3. Elija una API HTTP.
4. En la pestaña Monitor (Monitorear), en el panel de navegación principal, elija Logging (Registro).
5. Seleccione una etapa para activar el registro y elija Select (Seleccionar).

6. Elija Edit (Editar) para activar el registro de acceso.
7. Active el Access logging (Registro de acceso), ingrese un registro de CloudWatch y seleccione un formato de registro.
8. Seleccione Save.

AWS CLI

El siguiente comando create-log-group permite crear un grupo de registro:

```
aws logs create-log-group --log-group-name my-log-group
```

Necesita el nombre de recurso de Amazon (ARN) del grupo de registro para activar el registro. El formato del ARN es arn:aws:logs:*región*:*id de cuenta*:grupo de registro:*nombre del grupo de registro*.

El siguiente comando update-stage activa el registro para la etapa `$default` de una API HTTP.

```
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"}'
```

Ejemplo de formatos de registro

La consola de API Gateway incluye algunos ejemplos de los formatos de registro de acceso habituales, los cuales se detallan a continuación.

- `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 (valores separados por comas):

```
$context.identity.sourceIp,$context.requestTime,$context.httpMethod,
$context.routeKey,$context.protocol,$context.status,$context.responseLength,
$context.requestId,$context.extendedRequestId
```

Personalización de registros de acceso de las API de HTTP

Puede utilizar las siguientes variables para personalizar los registros de acceso de las API HTTP. Para obtener más información acerca de los registros de acceso para las API HTTP, consulte Configuración del registro de API HTTP en API Gateway.

Parámetro	Descripción
`$context.accountId`	El ID de cuenta de AWS del propietario de la API.
`$context.apiId`	El identificador que API Gateway asigna a su API.
`$context.authorizer.claims.` *property*	Una propiedad de las reclamaciones devueltas por el JSON Web Token (JWT) una vez que se ha autenticado correctamente al intermediario del método, como `$context.authorizer.claims.username` . Para obtener más información, consulte Control del acceso

Parámetro	Descripción
	a API HTTP con autorizadores de JWT en API Gateway. **Note** La llamada a `$context.authorizer.claims` devuelve null.
`$context.authorizer.error`	El mensaje de error devuelto por un autorizador.
`$context.authorizer.`*property*	El valor del par clave-valor especificado de la asignación `context` devuelto de una función de autorizador de Lambda para API Gateway. Por ejemplo, si el autorizador devuelve la siguiente asignación de `context`: `"context" : { "key": "value", "numKey": 1, "boolKey": true }` la llamada a `$context.authorizer.key` devuelve la cadena `"value"`, la llamada a `$context.authorizer.numKey` devuelve `1` y la llamada a `$context.authorizer.boolKey` devuelve `true`.
`$context.awsEndpointRequestId`	El ID de la solicitud del punto de enlace de AWS desde el encabezado `x-amz-request-id` o `x-amzn-requestId` .
`$context.awsEndpointRequestId2`	El ID de la solicitud del punto de enlace de AWS desde el encabezado `x-amz-id-2` .

Parámetro	Descripción
`$context.customDomain.basePathMatched`	La ruta de un mapeo de la API con la que coincidió una solicitud entrante. Aplicable cuando un cliente utiliza un nombre de dominio personalizado para acceder a una API. Por ejemplo, si un cliente envía una solicitud a `https://api.example.com/v1/orders/1234` y la solicitud empareja el mapeo de la API con la ruta `v1/orders`, el valor es `v1/orders`. Para obtener más información, consulte the section called "Asignación de API".
`$context.dataProcessed`	La cantidad de datos procesados en bytes.
`$context.domainName`	El nombre de dominio completo que se utiliza para invocar la API. Este debe ser el mismo que el encabezado `Host` entrante.
`$context.domainPrefix`	La primera etiqueta del `$context.domainName`.
`$context.error.message`	Una cadena que contiene un mensaje de error de API Gateway.
`$context.error.messageString`	El valor entrecomillado de `$context.error.message`, es decir, `"$context.error.message"`.
`$context.error.responseType`	Un tipo de `GatewayResponse`. Para obtener más información, consulte the section called "Métricas" y the section called "Configuración de respuestas de gateway para personalizar respuestas de errores".
`$context.extendedRequestId`	Es igual que `$context.requestId`.

Parámetro	Descripción
`$context.httpMethod`	El método HTTP utilizado. Los valores válidos son: `DELETE`, `GET`, `HEAD`, `OPTIONS`, `PATCH`, `POST` y `PUT`.
`$context.identity.accountId`	El ID de cuenta de AWS asociado con la solicitud. Compatible con rutas que utilizan la autorización de IAM.
`$context.identity.caller`	El identificador principal del intermediario que firmó la solicitud. Compatible con rutas que utilizan la autorización de IAM.
`$context.identity.cognitoAuthenticationProvider`	Una lista separada por comas de todos los proveedores de autenticación de Amazon Cognito utilizados por el intermediario que realiza la solicitud. Solo está disponible si la solicitud se firmó con las credenciales de Amazon Cognito. Por ejemplo, para una identidad de un grupo de usuarios de Amazon Cognito, `cognito-idp.`*`region`*`.amazonaws.com/`*`user_pool_id`*`,cognito-idp.`*`region`*`.amazonaws.com/`*`user_pool_id`*`:CognitoSignIn:`*`token subject claim`* Consulte Uso de las identidades federadas en la Guía para desarrolladores de Amazon Cognito para obtener información sobre los proveedores de autenticación de Amazon Cognito disponibles.

Parámetro	Descripción
`$context.identity.cognitoAuthenticationType`	El tipo de autenticación de Amazon Cognito del intermediario que realiza la solicitud. Solo está disponible si la solicitud se firmó con las credenciales de Amazon Cognito. Los valores posibles incluyen `authenticated` para identidades autenticadas y `unauthenticated` para identidades no autenticadas.
`$context.identity.cognitoIdentityId`	El ID de identidad de Amazon Cognito del intermediario que realiza la solicitud. Solo está disponible si la solicitud se firmó con las credenciales de Amazon Cognito.
`$context.identity.cognitoIdentityPoolId`	El ID del grupo de identidades de Amazon Cognito del intermediario que realiza la solicitud. Solo está disponible si la solicitud se firmó con las credenciales de Amazon Cognito.
`$context.identity.principalOrgId`	El ID de organización de AWS. Compatible con rutas que utilizan la autorización de IAM.
`$context.identity.clientCert.clientCertPem`	El certificado de cliente codificado en PEM que el cliente presentó durante la autenticación TLS mutua. Presente cuando un cliente accede a una API mediante un nombre de dominio personalizado que tiene una TLS mutua habilitada.
`$context.identity.clientCert.subjectDN`	Nombre distintivo del asunto del certificado que presenta un cliente. Presente cuando un cliente accede a una API mediante un nombre de dominio personalizado que tiene una TLS mutua habilitada.

Parámetro	Descripción
`$context.identity.clientCert.issuerDN`	Nombre distintivo del emisor del certificado que presenta un cliente. Presente cuando un cliente accede a una API mediante un nombre de dominio personalizado que tiene una TLS mutua habilitada.
`$context.identity.clientCert.serialNumber`	Número de serie del certificado. Presente cuando un cliente accede a una API mediante un nombre de dominio personalizado que tiene una TLS mutua habilitada.
`$context.identity.clientCert.validity.notBefore`	Fecha antes de la cual el certificado no es válido. Presente cuando un cliente accede a una API mediante un nombre de dominio personalizado que tiene una TLS mutua habilitada.
`$context.identity.clientCert.validity.notAfter`	Fecha después de la cual el certificado no es válido. Presente cuando un cliente accede a una API mediante un nombre de dominio personalizado que tiene una TLS mutua habilitada.
`$context.identity.sourceIp`	La dirección IP de origen de la conexión TCP inmediata que realiza la solicitud al punto de enlace de API Gateway.
`$context.identity.user`	El identificador principal del usuario que se autorizará a acceder al recurso. Compatible con rutas que utilizan la autorización de IAM.
`$context.identity.userAgent`	El encabezado `User-Agent` del intermediario de la API.

Parámetro	Descripción
`$context.identity.userArn`	El Nombre de recurso de Amazon (ARN) del usuario identificado después de la autentica ción. Compatible con rutas que utilizan la autorización de IAM. Para obtener más información, consulte https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html.
`$context.integration.error`	El mensaje de error devuelto por una integraci ón. Es igual que `$context.integrationErrorMessage` .
`$context.integration.integrationStatus`	Para la integración de proxy de Lambda, el código de estado que se devuelve desde AWS Lambda, y no desde el código de función de Lambda del backend.
`$context.integration.latency`	La latencia de integración en ms. Es igual que `$context.integrationLatency` .
`$context.integration.requestId`	El ID de solicitud del punto de conexión de AWS. Es igual que `$context.awsEndpointRequestId` .
`$context.integration.status`	El código de estado devuelto por una integraci ón. Para integraciones de proxy de Lambda, este es el código de estado que devuelve su código de la función de Lambda.
`$context.integrationErrorMessage`	Una cadena que contiene un mensaje de error de integración.
`$context.integrationLatency`	La latencia de integración en ms.

Parámetro	Descripción
`$context.integrationStatus`	Para la integración de proxy de Lambda, este parámetro representa el código de estado que se devuelve desde AWS Lambda, y no desde la función de Lambda del backend.
`$context.path`	Ruta de acceso de la solicitud. Por ejemplo, `/{stage}/root/child` .
`$context.protocol`	Protocolo de la solicitud; por ejemplo, `HTTP/1.1`. **Note** Las API de API Gateway pueden aceptar solicitudes HTTP/2, pero API Gateway envía solicitudes a las integraciones de backend mediante HTTP/1.1. Como resultado, el protocolo de solicitud se registra como HTTP/1.1 incluso si un cliente envía una solicitud que usa HTTP/2.
`$context.requestId`	El ID que API Gateway asigna a la solicitud de API.
`$context.requestTime`	Hora de la solicitud en formato CLF-(`dd/MMM/yyyy:HH:mm:ss +-hhmm`).
`$context.requestTimeEpoch`	Hora de la solicitud en formato Epoch.
`$context.responseLatency`	La latencia de respuesta en ms.
`$context.responseLength`	La duración de la carga de respuesta en bytes.
`$context.routeKey`	La clave de ruta de la solicitud de la API, por ejemplo, `/pets`.

Parámetro	Descripción
`$context.stage`	La etapa de implementación de la solicitud de la API (por ejemplo, `beta` o `prod`).
`$context.status`	El estado de respuesta de método.

Solución de problemas con las API de HTTP en API Gateway

Los siguientes temas le proporcionan consejos para solucionar errores y problemas que puedan surgir al utilizar API HTTP.

Temas

- Solución de problemas con las integraciones de Lambda para API HTTP
- Solución de problemas con los autorizadores JWT de la API HTTP

Solución de problemas con las integraciones de Lambda para API HTTP

A continuación se proporcionan consejos para solucionar errores y problemas que puedan surgir al utilizar AWS LambdaIntegraciones de con API HTTP.

Problema: mi API con una integración de Lambda devuelve `{"message":"Internal Server Error"}`

Para solucionar el error interno del servidor, agregue la variable de registro `$context.integrationErrorMessage` al formato de registro y vea sus registros de API HTTP. Para ello, haga lo siguiente:

Para crear un grupo de registros a través de la Consola de administración de AWS

1. Abra la consola de CloudWatch en https://console.aws.amazon.com/cloudwatch/.
2. Elija Log groups (Grupos de registros).
3. Elija Create log group (Crear grupo de registros).
4. Escriba un nombre de grupo de registros y, a continuación, elija Create (Crear).

5. Anote el nombre de recurso de Amazon (ARN) de su grupo de registros. El formato ARN es arn:aws:logs: *region*: *account-id*:log-group:*log-group-name*. Necesita el ARN del grupo de registros para habilitar el registro de acceso a su API HTTP.

Para agregar la variable de registro **`$context.integrationErrorMessage`**

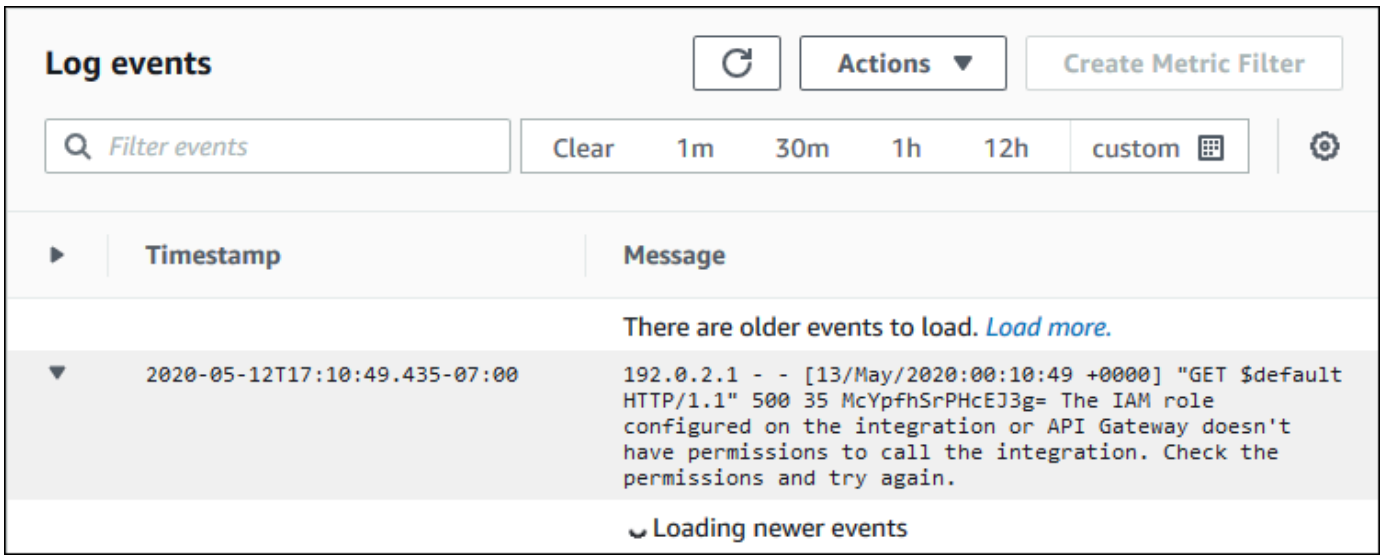
1. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
2. Elija su API HTTP.
3. En Monitor (Monitorización), elija Logging (Registro).
4. Seleccione una etapa de su API.
5. Elija Edit (Editar) y, a continuación, habilite el registro de acceso.
6. En Log destination (Destino del registro), introduzca el ARN del grupo de registros que creó en el paso anterior.
7. En Formato de registro, elija CLF. API Gateway crea un formato de registro de ejemplo.
8. Agregue `$context.integrationErrorMessage` al final del formato de registro.
9. Seleccione Guardar.

Para ver los registros de la API

1. Generación de registros. Utilice un navegador o `curl` para invocar su API.

```
$curl https://api-id.execute-api.us-west-2.amazonaws.com/route
```

2. Inicie sesión en la consola de API Gateway en https://console.aws.amazon.com/apigateway.
3. Elija su API HTTP.
4. En Monitorización, elija Registro.
5. Seleccione la etapa de la API para la cual habilitó el registro.
6. Elija View logs in CloudWatch (Ver registros en CloudWatch).
7. Elija la última secuencia de registro para ver sus registros de API HTTP.
8. Su entrada de registro debe tener un aspecto similar a este:



Dado que agregamos `$context.integrationErrorMessage` al formato de registro, vemos un mensaje de error en nuestros registros que resume el problema.

Los registros pueden incluir un mensaje de error diferente que indica que hay un problema con el código de función de Lambda. En ese caso, verifique su código de función de Lambda y compruebe que su función de Lambda devuelva una respuesta en el formato necesario. Si los registros no incluyen un mensaje de error, agregue `$context.error.message` y `$context.error.responseType` al formato de registro para obtener más información que le ayude a solucionar problemas.

En este caso, los registros muestran que API Gateway no tenía los permisos necesarios para invocar la función de Lambda.

Cuando crea una integración de Lambda en la consola de API Gateway, API Gateway configura automáticamente los permisos para invocar la función de Lambda. Cuando crea una integración de Lambda mediante la AWS CLI, CloudFormation o un SDK, debe conceder permisos para que API Gateway pueda invocar la función. Los siguientes comandos add-permission conceden permiso para diferentes rutas de API HTTP para invocar una función de Lambda.

Example Ejemplo: para la etapa **`$default`** y la ruta **`$default`** de una API HTTP

```
aws lambda add-permission \
    --function-name my-function \
    --statement-id apigateway-invoke-permissions \
    --action lambda:InvokeFunction \
    --principal apigateway.amazonaws.com \
```

```
    --source-arn "arn:aws:execute-api:us-west-2:123456789012:api-id/\$default/\
$default"
```

Example Ejemplo: para la etapa **`prod`** y la ruta **`test`** de una API HTTP

```
aws lambda add-permission \
    --function-name my-function \
    --statement-id apigateway-invoke-permissions \
    --action lambda:InvokeFunction \
    --principal apigateway.amazonaws.com \
    --source-arn "arn:aws:execute-api:us-west-2:123456789012:api-id/prod/*/test"
```

Confirme la política de la función en la pestaña Permisos de la consola de Lambda.

Intente volver a invocar su API. Debería ver la respuesta de su función de Lambda.

Solución de problemas con los autorizadores JWT de la API HTTP

A continuación se le proporcionan consejos para solucionar errores y problemas que puedan surgir al utilizar los autorizadores de JSON Web Token (JWT) con API HTTP.

Problema: mi API devuelve **`401 {"message":"Unauthorized"}`**

Compruebe el encabezado `www-authenticate` en la respuesta de la API.

El siguiente comando utiliza `curl` para enviar una solicitud a una API con un autorizador de JWT que utiliza `$request.header.Authorization` como su origen de identidad.

```
$curl -v -H "Authorization: token" https://api-id.execute-api.us-
west-2.amazonaws.com/route
```

La respuesta de la API incluye un encabezado `www-authenticate`.

```
...
< 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"}}
```

En este caso, el encabezado `www-authenticate` muestra que el token no se emitió para un destinatario válido. Para que API Gateway autorice una solicitud, la reclamación `aud` o `client_id` de JWT debe coincidir con una de las entradas de destinatario configuradas para el autorizador. API Gateway valida `client_id` solo si `aud` no está presente. Cuando `aud` y `client_id` están presentes, API Gateway evalúa `aud`.

También puede decodificar un JWT y comprobar que coincida con el emisor, el destinatario y los ámbitos que requiere su API. El sitio web jwt.io puede depurar JWT en el navegador. OpenID Foundation mantiene una lista de bibliotecas para trabajar con JWT.

Para obtener más información acerca de los autorizadores de JWT, consulte Control del acceso a API HTTP con autorizadores de JWT en API Gateway.

API de WebSocket de API Gateway

Una API de WebSocket en API Gateway es una colección de rutas de WebSocket que se integran con puntos de enlace de HTTP del backend, funciones de Lambda u otros servicios de AWS. Puede utilizar características de API Gateway para ayudarle con todos los aspectos del ciclo de vida de la API, desde la creación hasta el monitoreo de sus API de producción.

Las API de WebSocket en API Gateway son bidireccionales. Un cliente puede enviar mensajes a un servicio y los servicios pueden enviar mensajes a los clientes de forma independiente. Este comportamiento bidireccional permite interacciones más ricas entre clientes y servicios, ya que los servicios pueden enviar datos a los clientes sin requerir que los clientes realicen una solicitud explícita. Las API de WebSocket se utilizan a menudo en aplicaciones en tiempo real como aplicaciones de chat, plataformas de colaboración, juegos multijugador y plataformas de negociación financiera.

Para ver una aplicación de ejemplo con la que empezar, consulte Tutorial: Creación de una aplicación de chat de WebSocket con una API de WebSocket, Lambda y DynamoDB.

En esta sección, aprenderá a desarrollar, publicar, proteger y monitorear las API de WebSocket mediante API Gateway.

Temas

- Descripción general de las API de WebSocket en API Gateway
- Desarrollo de las API de WebSocket en API Gateway
- Publicación de las API de WebSocket para que los clientes las invoquen
- Protección de las API de WebSocket en API Gateway
- Supervisión de las API de WebSocket en API Gateway

Descripción general de las API de WebSocket en API Gateway

En API Gateway puede crear una API de WebSocket como un frontend con estado para un servicio de AWS (como Lambda o DynamoDB) o para un punto de enlace HTTP. La API de WebSocket invoca al backend en función del contenido de los mensajes que recibe de las aplicaciones cliente.

A diferencia de una API de REST, que recibe las solicitudes y responde a ellas, una API de WebSocket admite la comunicación bidireccional entre las aplicaciones cliente y el backend. El backend puede enviar mensajes de devolución de llamada a los clientes conectados.

En la API de WebSocket, los mensajes JSON entrantes se dirigen a las integraciones de backend en función de las rutas que se hayan configurado. (Los mensajes que no son de JSON se dirigen a la ruta `$default` que se configure).

Una ruta incluye una clave de ruta, que es el valor que se espera una vez que se evalúa una expresión de selección de ruta. `routeSelectionExpression` es un atributo definido en el nivel de API. Especifica una propiedad JSON cuya presencia se espera en la carga del mensaje. Para obtener más información sobre las expresiones de selección de ruta, consulte the section called “”.

Por ejemplo, si los mensajes JSON contienen una propiedad `action` y desea realizar diferentes acciones en función de esta propiedad, la expresión de selección de ruta podría ser `${request.body.action}`. La tabla de enrutamiento especificaría la acción que se debe realizar comparando el valor de la propiedad `action` con los valores de clave de ruta personalizados que se han definido en la tabla.

Uso de rutas para una API de WebSocket

Existen tres rutas predefinidas que se pueden utilizar: `$connect`, `$disconnect` y `$default`. Además, es posible crear rutas personalizadas.

- API Gateway llama a la ruta `$connect` al iniciarse una conexión persistente entre el cliente y una API de WebSocket.
- API Gateway llama a la ruta `$disconnect` cuando el cliente o el servidor se desconecta de la API.
- API Gateway llama a una ruta personalizada después de evaluar la expresión de selección de ruta con respecto al mensaje si se encuentra una ruta coincidente; esta coincidencia determina qué integración se invoca.
- API Gateway llama a la ruta `$default` si la expresión de selección de ruta no puede evaluarse con respecto al mensaje o no se encuentra ninguna ruta coincidente.

Para obtener más información sobre las rutas `$connect` y `$disconnect`, consulte the section called “Administración de usuarios conectados y aplicaciones cliente”.

Para obtener más información sobre la ruta `$default` y las rutas personalizadas, consulte the section called “Invocación de la integración del backend”.

Envío de datos a las aplicaciones cliente conectadas

Los servicios de backend pueden enviar datos a las aplicaciones cliente conectadas. Puede enviar datos si hace lo siguiente:

- Use una integración puede enviar una respuesta, que se devuelve al cliente mediante una respuesta de ruta que usted haya definido.
- Puede utilizar la API `@connections` para enviar una solicitud POST. Para obtener más información, consulte the section called “Uso de comandos de `@connections` en el servicio de backend”.

Códigos de estado de la API de WebSocket

Las API de WebSocket de API Gateway utilizan los siguientes códigos de estado para la comunicación del servidor al cliente, tal y como se describe en el registro de números de códigos de cierre de WebSocket:

1001

API Gateway devuelve este código de estado cuando el cliente permanece inactivo durante 10 minutos o alcanza el tiempo de conexión máximo de 2 horas.

1003

API Gateway devuelve este código de estado cuando un punto de conexión recibe un tipo de medio binario. Los tipos de medios binarios no se admiten en las API de WebSocket.

1005

API Gateway devuelve este código de estado si el cliente envía una trama de cierre sin un código de cierre.

1006

API Gateway devuelve este código de estado si se produce un cierre inesperado de la conexión, como el cierre de la conexión TCP sin una trama de cierre de WebSocket.

1008

API Gateway devuelve este código de estado cuando un punto de conexión recibe demasiadas solicitudes de un cliente concreto.

1009

API Gateway devuelve este código de estado cuando un punto de conexión recibe un mensaje demasiado grande para procesarlo.

1011

API Gateway devuelve este código de estado cuando se produce un error interno del servidor.

1012

API Gateway devuelve este código de estado si el servicio se reinicia.

Administración de usuarios conectados y aplicaciones cliente: rutas `$connect` y `$disconnect`

En la siguiente sección se describe cómo utilizar las rutas `$connect` y `$disconnect` para la API de WebSocket.

Temas

- La ruta $connect
- Transmitir información de conexión de la ruta $connect
- La ruta $disconnect

La ruta `$connect`

Las aplicaciones cliente se conectan a la API de WebSocket enviando una solicitud de actualización de WebSocket. Si la solicitud se realiza correctamente, la ruta `$connect` se ejecuta mientras se establece la conexión.

Dado que la conexión de WebSocket es una conexión con estado, solo se puede configurar la autorización en la ruta `$connect`. `AuthN`/`AuthZ` solo se realizará en el momento de la conexión.

Hasta que se complete la ejecución de la integración asociada a la ruta `$connect`, la solicitud de actualización está pendiente y la conexión real no se establece. Si la solicitud `$connect` da error (por ejemplo, debido a un error de `AuthN`/`AuthZ` o de integración), la conexión no se realizará.

> **Note**
>
> Si se produce un error en la autorización en `$connect`, la conexión no se establece y el cliente recibirá una respuesta `401` o `403`.

La configuración de una integración para `$connect` es opcional. Considere la posibilidad de configurar una integración `$connect` si:

- Desea habilitar a los clientes para especificar subprotocolos mediante el campo `Sec-WebSocket-Protocol`. Para ver código de ejemplo, consulte Configuración de una ruta `$connect` que requiere un subprotocolo WebSocket.
- Desea recibir una notificación cuando los clientes se conectan.
- Desea limitar las conexiones o controlar quién se conecta.
- Desea que el backend envíe mensajes de respuesta a los clientes mediante una URL de devolución de llamada.
- Desea almacenar los ID de conexión, los nombres de dominio u otros detalles de conexión en una base de datos (por ejemplo, Amazon DynamoDB). El ID de conexión es una cadena de longitud variable cuya longitud y formato pueden cambiar.

Transmitir información de conexión de la ruta **`$connect`**

Puede utilizar integraciones con y sin proxy para transferir información de la ruta `$connect` a una base de datos u otro Servicio de AWS.

Para transferir información de conexión mediante una integración de proxy

En ese caso, puede acceder a la información de conexión desde una integración de proxy de Lambda. Utilice otro Servicio de AWS o función de AWS Lambda para publicar en la conexión.

La siguiente función de Lambda muestra cómo utilizar el objeto `requestContext` para registrar el ID de conexión, el nombre de dominio, el nombre de etapa y las cadenas de consulta.

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}
```

Para transferir información de conexión mediante una integración que no sea de proxy

- Puede acceder a la información de conexión con una integración que no sea de proxy. Configure la solicitud de integración y proporcione una plantilla de solicitud de API de WebSocket. La siguiente plantilla de mapeo Velocity Template Language (VTL) proporciona una solicitud de integración. Esta solicitud envía los siguientes detalles a una integración que no sea de proxy:
  - ID de la conexión
  - Nombre del dominio
  - Nombre de la fase
  - Ruta
  - Encabezados

- Cadenas de consulta

Esta solicitud envía el ID de conexión, el nombre de dominio, el nombre de etapa, las rutas, los encabezados y las cadenas de consulta a una integración que no sea de proxy.

```
{
    "connectionId": "$context.connectionId",
    "domain": "$context.domainName",
    "stage": "$context.stage",
    "params": "$input.params()"
}
```

Para obtener más información sobre la configuración de transformaciones de datos, consulte the section called “Transformaciones de datos”.

Para completar la solicitud de integración, establezca `StatusCode: 200` para la respuesta de integración. Para obtener más información sobre la configuración de una respuesta de integración, consulte Configuración de una respuesta de integración mediante la consola de API Gateway.

La ruta **`$disconnect`**

La ruta `$disconnect` se ejecuta una vez que se ha cerrado la conexión.

La conexión la puede cerrar el servidor o el cliente. Como la conexión ya está cerrada cuando se ejecuta, `$disconnect` es un evento de mejor esfuerzo. API Gateway hará todo lo posible para entregar el evento `$disconnect` a su integración, pero no puede garantizar la entrega.

El backend puede iniciar la desconexión mediante la API `@connections`. Para obtener más información, consulte the section called “Uso de comandos de `@connections` en el servicio de backend”.

Invocación de la integración de backend con la ruta **`$default`** y las rutas personalizadas en API Gateway

En la siguiente sección se describe cómo invocar su integración de backend mediante la ruta `$default` o una ruta personalizada para una API de WebSocket.

Temas

- Uso de rutas para procesar mensajes
- La ruta $default
- Rutas personalizadas
- Uso de las integraciones de API de WebSocket de API Gateway para conectarse a la lógica de negocio
- Diferencias importantes entre las API de WebSocket y las API de REST

Uso de rutas para procesar mensajes

En las API de WebSocket de API Gateway, es posible enviar mensajes desde el cliente al servicio de backend y viceversa. A diferencia del modelo solicitud/respuesta de HTTP, en WebSocket el backend puede enviar mensajes al cliente sin que este realice ninguna acción.

Los mensajes pueden ser tener el formato JSON u otro distinto. Sin embargo, solo los mensajes JSON se pueden direccionar a integraciones específicas en función del contenido del mensaje. Los mensajes que no tienen el formato JSON se transmiten a través del backend por la ruta `$default`.

> **Note**
>
> API Gateway admite cargas de mensajes de hasta 128 KB con un tamaño máximo de trama de 32 KB. Si un mensaje supera los 32 KB, se debe dividir en varias tramas, cada una con tamaño máximo de 32 KB. Si se recibe un mensaje (o una trama) más grande, la conexión se cierra con el código 1009.
> Actualmente, no se admiten cargas binarias. Si se recibe una trama binaria, la conexión se cierra con el código 1003. Sin embargo, es posible convertir cargas binarias a texto. Consulte the section called “Tipos de medios binarios”.

Con las API de WebSocket en API Gateway, los mensajes JSON se pueden direccionar para ejecutar un servicio de backend específico en función del contenido del mensaje. Cuando un cliente envía un mensaje a través de su conexión WebSocket, esto da lugar a una solicitud de ruta a la API de WebSocket. La solicitud se emparejará con la ruta que tenga la clave de ruta correspondiente en API Gateway. Puede configurar una solicitud de ruta para una API de WebSocket en la consola de API Gateway, mediante la AWS CLI o mediante un AWS SDK.

> **Note**
>
> Tanto la AWS CLI como los SDK de AWS le permiten crear rutas antes o después de generar las integraciones. Actualmente, la consola no admite la reutilización de integraciones, por lo que debe crear primero la ruta y, a continuación, la integración de dicha ruta.

Puede configurar API Gateway para que realice la validación en una solicitud de ruta antes de continuar con la solicitud de integración. Si no se supera la validación, API Gateway rechaza la solicitud sin llamar al backend, envía una respuesta de gateway `"Bad request body"` similar a la siguiente al cliente y publica los resultados de la validación en CloudWatch Logs:

```
{"message" : "Bad request body", "connectionId": "{connectionId}", "messageId":
 "{messageId}"}
```

Esto reduce las llamadas innecesarias al backend y le permite centrarse en los demás requisitos de la API.

También puede definir una respuesta de ruta para las rutas de la API con objeto de permitir la comunicación bidireccional. Una respuesta de ruta describe qué datos se enviarán al cliente al finalizar la integración de una ruta determinada. No es necesario definir una respuesta para una ruta si, por ejemplo, desea que un cliente envíe mensajes al backend sin recibir una respuesta (comunicación unidireccional). Sin embargo, si no se proporciona una respuesta de ruta, API Gateway no enviará ninguna información sobre el resultado de la integración a los clientes.

La ruta `$default`

Cada API de WebSocket de API Gateway puede tener una ruta `$default`. Se trata de un valor de direccionamiento especial que se puede utilizar de las siguientes formas:

- Puede utilizarlo junto con claves de ruta definidas para especificar una ruta "alternativa" (por ejemplo, una integración simulada genérica que devuelve un mensaje de error determinado) para los mensajes entrantes que no coinciden con ninguna de las claves de ruta definidas.
- Puede utilizarlo sin claves de ruta definidas para especificar un modelo de proxy que delega el direccionamiento en un componente del backend.
- También puede usarlo si desea especificar una ruta para las cargas que no son JSON.

Rutas personalizadas

Si desea invocar una integración específica en función del contenido del mensaje, puede hacerlo mediante la creación de una ruta personalizada.

Una ruta personalizada utiliza la clave de ruta y la integración que se especifiquen. Si un mensaje entrante contiene una propiedad JSON y dicha propiedad toma un valor que coincide con el valor de la clave de ruta, API Gateway invoca la integración. (Para obtener más información, consulte the section called “Descripción general de las API de WebSocket”.)

Por ejemplo, supongamos que desea crear una aplicación de sala de chat. Puede comenzar creando una API de WebSocket cuya expresión de selección de ruta sea `$request.body.action`. A continuación, puede definir dos rutas: `joinroom` y `sendmessage`. Una aplicación cliente podría invocar la ruta `joinroom` enviando un mensaje como el siguiente:

```
{"action":"joinroom","roomname":"developers"}
```

También podría invocar la ruta `sendmessage` enviando un mensaje como el siguiente:

```
{"action":"sendmessage","message":"Hello everyone"}
```

Uso de las integraciones de API de WebSocket de API Gateway para conectarse a la lógica de negocio

Después de configurar una ruta para una API de WebSocket de API Gateway, debe especificar la integración que desea utilizar. Al igual que ocurre con las rutas, que pueden tener una solicitud de ruta y una respuesta de ruta, las integraciones pueden tener una solicitud de integración y una respuesta de integración. Una solicitud de integración contiene la información que espera el backend para poder procesar la solicitud procedente del cliente. Una respuesta de integración contiene los datos que el backend devuelve a API Gateway y que se pueden utilizar para construir un mensaje que se envía al cliente (si se ha definido una respuesta de ruta).

Para obtener más información sobre la configuración de integraciones, consulte the section called “Integraciones”.

Diferencias importantes entre las API de WebSocket y las API de REST

Las integraciones para las API de WebSocket son similares a las integraciones para las API de REST, con las siguientes salvedades:

- Actualmente, se debe crear primero una ruta en la consola de API Gateway y, a continuación, crear una integración como destino de dicha ruta. Sin embargo, en la API y la CLI, es posible crear rutas e integraciones de forma independiente, en cualquier orden.
- Puede utilizar una única integración para varias rutas. Por ejemplo, si dispone de un conjunto de acciones estrechamente relacionadas entre sí, podría ser conveniente que todas estas rutas vayan a una única función de Lambda. En lugar de definir los detalles de la integración varias veces, puede especificarla una vez y asignarla a cada una de las rutas relacionadas.

> **Note**
>
> Actualmente, la consola no admite la reutilización de integraciones, por lo que debe crear primero la ruta y, a continuación, la integración de dicha ruta.
> En la AWS CLI y los SDK de AWS, se puede reutilizar una integración estableciendo el destino de la ruta en el valor "`integrations/{integration-id}`", donde "`{integration-id}`" es el ID exclusivo de la integración que se va a asociar a la ruta.

- API Gateway ofrece varias expresiones de selección que se pueden utilizar en las rutas e integraciones. La selección de una plantilla de entrada o una asignación de salida no depende del tipo de contenido. Al igual que ocurre con las expresiones de selección de ruta, puede definir una expresión de selección que API Gateway evaluará para elegir el elemento correcto. Todas ellas recurrirán a la plantilla `$default` si no se encuentra una plantilla coincidente.
  - En las solicitudes de integración, la expresión de selección de plantillas admite `$request.body.<json_path_expression>` y valores estáticos.
  - En las respuestas de integración, la expresión de selección de plantillas admite `$request.body.<json_path_expression>`, `$integration.response.statuscode` y `$integration.response.header.<headerName>`, además de valores estáticos.

En el protocolo HTTP, en el que se envían solicitudes y respuestas de forma síncrona, la comunicación es básicamente unidireccional. En el protocolo WebSocket, la comunicación es bidireccional. Las respuestas son asíncronas y el cliente no las recibe necesariamente en el mismo orden en el que se enviaron sus mensajes. Además, el backend puede enviar mensajes al cliente.

> **Note**
>
> En una ruta que se ha configurado para utilizar la integración `AWS_PROXY` o `LAMBDA_PROXY`, la comunicación es unidireccional y API Gateway no pasará la respuesta del backend

a la respuesta de ruta de forma automática. Por ejemplo, en el caso de la integración `LAMBDA_PROXY`, el cuerpo que devuelve la función de Lambda no se enviará al cliente. Si desea que el cliente reciba respuestas de integración, debe definir una respuesta de ruta para posibilitar la comunicación bidireccional.

Expresiones de selección de WebSocket

API Gateway utiliza expresiones de selección como forma de evaluar el contexto de solicitud y respuesta y producir una clave. La clave se utiliza para seleccionar un valor de un conjunto de valores posibles, normalmente proporcionados por el desarrollador de la API. El conjunto exacto de variables admitidas variará en función de la expresión dada. A continuación se describe con más detalle cada expresión.

En todas las expresiones, el lenguaje sigue el mismo conjunto de reglas:

- Las variables tienen el prefijo `"$"`.
- Se pueden utilizar llaves para definir explícitamente los límites de las variables, por ejempl., `"${request.body.version}-beta"`.
- Se admite el uso de varias variables, pero la evaluación solo se produce una vez (sin evaluación recursiva).
- A los signos de dólar ($) se les puede aplicar una secuencia de escape con `"\"`. Esto resulta muy útil al definir una expresión que se asigna a la clave `$default` reservada, como, por ejemplo, `"\$default"`.
- En algunos casos, se requiere un patrón de formato. En este caso, la expresión debe encerrarse entre barras inclinadas (`"/"`), por ejemplo `"/2\d\d/"`, para que coincida con los códigos de estado 2*XX*.

Temas

- Expresiones de selección de respuesta de ruta
- Expresiones de selección de clave de API
- Expresiones de selección de asignación de API
- Resumen de expresiones de selección de WebSocket

Expresiones de selección de respuesta de ruta

Las respuestas de ruta se utilizan para modelar una respuesta desde el backend al cliente. En la API de WebSocket, la respuesta de ruta es opcional. Si se define, indica a API Gateway que debe devolver una respuesta a un cliente al recibir un mensaje de WebSocket.

La evaluación de la expresión de selección de respuesta de ruta produce una clave de respuesta de ruta. Con el tiempo, esta clave se utilizará para elegir una de las respuestas `RouteResponses` asociadas a la API. Sin embargo, actualmente solo se admite la clave `$default`.

Expresiones de selección de clave de API

Esta expresión se evalúa cuando el servicio determina que la solicitud dada debe continuar solo si el cliente proporciona una clave de API válida.

Actualmente, los únicos dos valores admitidos son `$request.header.x-api-key` y `$context.authorizer.usageIdentifierKey`.

Expresiones de selección de asignación de API

Esta expresión se evalúa para determinar qué etapa de API está seleccionada cuando se realiza una solicitud mediante un dominio personalizado.

Actualmente el único valor admitido es `$request.basepath`.

Resumen de expresiones de selección de WebSocket

En la tabla siguiente, se resumen los casos de uso de las expresiones de selección en las API de WebSocket:

Expresión de selección	Se evalúa como la clave para	Notas	Ejemplo de caso de uso
`Api.RouteSelectionExpression`	`Route.RouteKey`	`$defaul` se admite como una	Direccionar los mensajes de WebSocket

Expresión de selección	Se evalúa como la clave para	Notas	Ejemplo de caso de uso
		ruta catch-all.	en función del contexto de una solicitud de cliente.

Expresión de selección	Se evalúa como la clave para	Notas	Ejemplo de caso de uso
`Route.ModelSelectionExpression`	Clave para `Route.RequestModels`	Opcional Si se proporcio na para una integraci ón que no sea de proxy, se produce la validació n del modelo. `$defaul` se admite como método catch-all.	Realizar la validació n de solicitud es de forma dinámica dentro de la misma ruta.

Expresión de selección	Se evalúa como la clave para	Notas	Ejemplo de caso de uso
`Integration.TemplateSelectionExpression`	Clave para `Integration.RequestTemplates`	Opcional Se puede proporcionar para la integración que no sea de proxy con el fin de manipular las cargas entrantes. `${request.body.jsonPath}` admiten y valores estáticos. `$default` se admite	Manipular la solicitud del intermediario en función de las propiedades dinámicas de la solicitud.

Expresión de selección	Se evalúa como la clave para	Notas	Ejemplo de caso de uso
		como método catch-all.	

Expresión de selección	Se evalúa como la clave para	Notas	Ejemplo de caso de uso
`Integration.IntegrationResponseSelectionExpression`	`IntegrationResponse.IntegrationResponseKey`	Opcional. Se puede proporcionar para una integración que no sea de proxy. Actúa como una coincidencia de patrones para los mensajes de error (de Lambda) o los códigos de estado (de las integraci	Manipular la respuesta del backend. Elegir la acción que debe realizarse en función de la respuesta dinámica del backend (por ejemplo, controlar de forma inequívoca determinados errores).

Expresión de selección	Se evalúa como la clave para	Notas	Ejemplo de caso de uso
		ones HTTP). `$defaul` es necesari en las integraci ones que no sean de proxy para que actúe como el método catch-all para las respuest s correctas.	

Expresión de selección	Se evalúa como la clave para	Notas	Ejemplo de caso de uso
`IntegrationResponse.TemplateSelectionExpression`	Clave para `IntegrationResponse.ResponseTemplates`	Opcional Se puede proporcionar para una integración que no sea de proxy. Se admite $default.	En algunos casos, una propiedad dinámica de la respuesta puede dictar la necesidad de transformaciones diferentes dentro de la misma ruta y la integración asociada. `${request.body.jsonPath}` , `${integration.response.stat`

Expresión de selección	Se evalúa como la clave para	Notas	Ejemplo de caso de uso
			`uscode}` , `${integration.response.header.headerName}` , `${integration.response.multivalueheader.headerName}` , Se admiten , , , y valores estáticos . `$default` se admite como método catch-all.

Expresión de selección	Se evalúa como la clave para	Notas	Ejemplo de caso de uso
`Route.RouteResponseSelectionExpression`	`RouteResponse.RouteResponseKey`	Debe proporcionarse para iniciar la comunicación bidireccional en una ruta de WebSocket. Actualmente, este valor está restringido únicamente a `$default`.	
`RouteResponse.ModelSelectionExpression`	Clave para `RouteResponse.RequestModels`	No se admite actualmente.	

Desarrollo de las API de WebSocket en API Gateway

En esta sección se proporciona información detallada acerca de las capacidades de API Gateway que necesitará para desarrollar las API de API Gateway.

A medida que se desarrolla la API de API Gateway, se decide sobre una serie de características de la API. Estas características dependen del uso de la API. Por ejemplo, es posible que quiera permitir solo a ciertos clientes llamar a la API o puede que quiera que esté disponible para todos. Quizás desee que una llamada a la API ejecute una función de Lambda, haga una consulta a la base de datos o llame a una aplicación.

Temas

- Creación de las API de WebSocket en API Gateway
- Tipos de direcciones IP para las API de WebSocket en API Gateway
- Creación de rutas para las API de WebSocket en API Gateway
- Control y administración del acceso a las API de WebSocket en API Gateway
- Integraciones de API de WebSocket en API Gateway
- Solicitud de la validación de las API de WebSocket en API Gateway
- Transformaciones de datos para las API de WebSocket en API Gateway
- Tipos de medios binarios para las API de WebSocket en API Gateway
- Invocación de las API de WebSocket

Creación de las API de WebSocket en API Gateway

Puede crear una API de WebSocket en la consola de API Gateway mediante el comando create-api de la AWS CLI o mediante el comando `CreateApi` en un AWS SDK. Los procedimientos que se describen a continuación muestran cómo crear una API de WebSocket nueva.

> **Note**
>
> Las API de WebSocket solo son compatibles con TLS 1.2 y TLS 1.3. No se admiten versiones de TLS anteriores.

Creación de una API de WebSocket mediante comandos de la AWS CLI

El siguiente comando create-api permite crear una API con la expresión de selección de ruta `$request.body.action`:

```
aws apigatewayv2 --region us-east-1 create-api --name "myWebSocketApi3" --protocol-type
 WEBSOCKET --route-selection-expression '$request.body.action'
```

El resultado es similar al siguiente:

```
{
    "ApiKeySelectionExpression": "$request.header.x-api-key",
    "Name": "myWebSocketApi3",
    "CreatedDate": "2018-11-15T06:23:51Z",
    "ProtocolType": "WEBSOCKET",
    "RouteSelectionExpression": "'$request.body.action'",
    "ApiId": "aabbccddee"
}
```

Creación de una API de WebSocket mediante la consola de API Gateway

Para crear una API de WebSocket en la consola, elija el protocolo WebSocket y asígnele un nombre a la API.

> **Important**
>
> Una vez que haya creado la API, no podrá cambiar el protocolo que haya elegido para ella. No hay forma de convertir una API de WebSocket en una API de REST o viceversa.

Para crear una API de WebSocket mediante la consola de API Gateway

1. Inicie sesión en la consola de API Gateway y elija Create API (Crear API).
2. En WebSocket API (API de WebSocket), elija Build (Generar). Solo se admiten puntos de conexión regionales.
3. En Nombre de API, escriba el nombre de la API.
4. En Expresión de selección de ruta, introduzca un valor. Por ejemplo, `$request.body.action`.

   Para obtener más información sobre las expresiones de selección de ruta, consulte the section called “”.

5. Realice una de las siguientes acciones:
   - Elija Crear una API en blanco para crear una API sin rutas.
   - Seleccione Siguiente para asociar rutas a su API.

   Puede asociar rutas después de crear su API.

Tipos de direcciones IP para las API de WebSocket en API Gateway

Al crear una API, se especifica el tipo de direcciones IP que puede invocar la API. Puede elegir IPv4 para resolver las direcciones IPv4 para invocar la API o puede elegir pila doble para permitir que las direcciones IPv4 e IPv6 invoquen la API. Le recomendamos que establezca el tipo de dirección IP en pila doble para mitigar el agotamiento del espacio IP o para mejorar la postura de seguridad. Para obtener más información sobre los beneficios de un tipo de dirección IP de pila doble, consulte IPv6 en AWS.

Consideraciones para los tipos de direcciones IP

Es posible que las siguientes consideraciones afecten al uso de los tipos de direcciones IP:

- El tipo de dirección IP predeterminado para todas las API de WebSocket es IPv4.
- Si cambia el tipo de dirección IP de una API existente de IPv4 a pila doble, confirme que todas las políticas que controlan el acceso a las API se hayan actualizado para tener en cuenta las llamadas de IPv6. Cuando cambia el tipo de dirección IP, el cambio surte efecto inmediatamente.
- La API se puede asignar a un nombre de dominio personalizado con un tipo de dirección IP diferente al de la API. Si desactiva el punto de conexión de la API predeterminado, esto podría afectar a la forma en que los intermediarios pueden invocar a la API.

Cambio del tipo de dirección IP de una API de WebSocket

Puede cambiar el tipo de dirección IP actualizando la configuración de la API. Puede actualizar la configuración de la API mediante el uso de la Consola de administración de AWS, la AWS CLI, CloudFormation o un AWS SDK. Si cambia el tipo de dirección IP de la API, no es necesario volver a implementar la API para que los cambios surtan efecto.

Consola de administración de AWS

Cambio del tipo de dirección IP de una API de WebSocket

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Elija una API de WebSocket.
3. Elija Configuración de la API y, a continuación, elija Editar.
4. Para tipo de dirección IP, seleccione IPv4 o Pila doble.
5. Seleccione Save.

   El cambio en la configuración de la API se aplicará de forma inmediata.

AWS CLI

El siguiente comando update-api actualiza una API para que tenga un tipo de dirección IP de pila doble:

```
aws apigatewayv2 update-api \
    --api-id abcd1234 \
    --ip-address-type dualstack
```

El resultado será similar al siguiente:

```
{
    "ApiEndpoint": "https://abcd1234.execute-api.us-east-1.amazonaws.com",
    "ApiId": "abcd1234",
    "ApiKeySelectionExpression": "$request.header.x-api-key",
    "CreatedDate": "2025-02-04T22:20:20+00:00",
    "DisableExecuteApiEndpoint": false,
    "Name": "My-WebSocket-API",
    "ProtocolType": "WEBSOCKET",
    "RouteSelectionExpression": "$request.method $request.path",
    "Tags": {},
    "NotificationUris": [],
    "IpAddressType": "dualstack"
}
```

Creación de rutas para las API de WebSocket en API Gateway

En la API de WebSocket, los mensajes JSON entrantes se dirigen a las integraciones de backend en función de las rutas que se hayan configurado. (Los mensajes que no son de JSON se dirigen a la ruta `$default` que se configure).

Una ruta incluye una clave de ruta, que es el valor que se espera una vez que se evalúa una expresión de selección de ruta. `routeSelectionExpression` es un atributo definido en el nivel de API. Especifica una propiedad JSON cuya presencia se espera en la carga del mensaje. Para obtener más información sobre las expresiones de selección de ruta, consulte the section called “”.

Por ejemplo, si los mensajes JSON contienen una propiedad `action` y desea realizar diferentes acciones en función de esta propiedad, la expresión de selección de ruta podría ser `${request.body.action}`. La tabla de enrutamiento especificaría la acción que se debe realizar comparando el valor de la propiedad `action` con los valores de clave de ruta personalizados que se han definido en la tabla.

Existen tres rutas predefinidas que se pueden utilizar: `$connect`, `$disconnect` y `$default`. Además, es posible crear rutas personalizadas.

- API Gateway llama a la ruta `$connect` al iniciarse una conexión persistente entre el cliente y una API de WebSocket.
- API Gateway llama a la ruta `$disconnect` cuando el cliente o el servidor se desconecta de la API.
- API Gateway llama a una ruta personalizada después de evaluar la expresión de selección de ruta con respecto al mensaje si se encuentra una ruta coincidente; esta coincidencia determina qué integración se invoca.
- API Gateway llama a la ruta `$default` si la expresión de selección de ruta no puede evaluarse con respecto al mensaje o no se encuentra ninguna ruta coincidente.

Expresiones de selección de ruta

Una expresión de selección de ruta se evalúa cuando el servicio está seleccionando la ruta que debe seguir un mensaje entrante. El servicio utiliza la ruta cuya `routeKey` coincida exactamente con el valor evaluado. Si no hay ninguna coincidencia y existe una ruta con la clave de ruta `$default`, se selecciona esta. Si no hay rutas que coincidan con el valor evaluado y no existe ninguna ruta `$default`, el servicio devuelve un error. En las API basadas en WebSocket, la expresión debe tener el formato `$request.body.`*`{path_to_body_element}`*.

Por ejemplo, supongamos que está enviando el siguiente mensaje JSON:

```
{
    "service" : "chat",
    "action" : "join",
    "data" : {
        "room" : "room1234"
   }
}
```

Es posible que desee seleccionar el comportamiento de la API en función de la propiedad `action`. En ese caso, podría definir la siguiente expresión de selección de ruta:

```
$request.body.action
```

En este ejemplo, `request.body` hace referencia a la carga JSON del mensaje y `.action` es una expresión JSONPath. Puede utilizar cualquier expresión de ruta JSON después de `request.body`, pero tenga en cuenta que el resultado se representará en forma de cadena. Por ejemplo, si la expresión JSONPath devuelve una matriz de dos elementos, esta se presentará como la cadena `"[item1, item2]"`. Por este motivo, es conveniente que la expresión dé como resultado un valor y no una matriz ni un objeto.

Puede utilizar simplemente un valor estático o puede utilizar varias variables. En la tabla siguiente, se muestran ejemplos y sus resultados evaluados frente a la carga anterior.

Expresión	Resultado evaluado	Descripción
`$request.body.action`	`join`	Una variable desencapsulada
`${request.body.action}`	`join`	Una variable encapsulada

Expresión	Resultado evaluado	Descripción
`${request.body.service}/${request.body.action}`	`chat/join`	Varias variables con valores estáticos
`${request.body.action}-${request.body.invalidPath}`	`join-`	Si no se encuentra JSONPath, la variable se resueve como "".
`action`	`action`	Valor estático
`\$default`	`$default`	Valor estático

El resultado evaluado se utiliza para encontrar una ruta. Si hay una ruta con una clave de ruta coincidente, se selecciona la ruta para procesar el mensaje. Si no se encuentra ninguna ruta coincidente, API Gateway intenta encontrar la ruta `$default` si está disponible. Si no se ha definido la ruta `$default`, API Gateway devuelve un error.

Configuración de rutas para una API de WebSocket en API Gateway

La primera vez que se crea una API de WebSocket, existen tres rutas predefinidas: `$connect`, `$disconnect` y `$default`. Puede crearlas mediante la consola, la API o la AWS CLI. Si lo desea, puede crear rutas personalizadas. Para obtener más información, consulte the section called “Descripción general de las API de WebSocket”.

> **Note**
>
> En la CLI, puede crear las rutas antes o después de crear las integraciones y puede volver a utilizar la misma integración para varias rutas.

Creación de una ruta mediante la consola de API Gateway

Para crear una ruta mediante la consola de API Gateway

1. Inicie sesión en la consola de API Gateway, elija la API y, a continuación, elija Routes (Rutas).
2. Elija Crear ruta.
3. En Clave de la ruta, ingrese el nombre de la clave de la ruta. Puede crear las rutas predefinidas (`$connect`, `$disconnect` y `$default`) o una ruta personalizada.

   > **Note**
   >
   > Cuando cree una ruta personalizada, no utilice el prefijo $ en el nombre de la clave de ruta. Este prefijo está reservado para las rutas predefinidas.

4. Seleccione y configure el tipo de integración de la ruta. Para obtener más información, consulte the section called “Configuración de una solicitud de integración de la API de WebSocket mediante la consola de API Gateway”.

Creación de una ruta utilizando la AWS CLI

El siguiente comando create-route permite crear una ruta:

```
aws apigatewayv2 --region us-east-1 create-route --api-id aabbccddee --route-key
 $default
```

El resultado será similar al siguiente:

```
{
    "ApiKeyRequired": false,
    "AuthorizationType": "NONE",
    "RouteKey": "$default",
    "RouteId": "1122334"
}
```

Especificación de los ajustes de la solicitud de ruta para **$connect**

Cuando se configura la ruta `$connect` para la API, están disponibles los siguientes ajustes opcionales para permitir la autorización para la API. Para obtener más información, consulte the section called “La ruta $connect”.

- Authorization (Autorización): si no se necesita autorización, puede especificar `NONE`. De lo contrario, puede especificar:
  - `AWS_IAM` para utilizar políticas estándar de AWS IAM con el fin de controlar el acceso a la API.
  - `CUSTOM` para implementar la autorización para una API mediante la especificación de una función de autorizador de Lambda que se ha creado previamente. El autorizador puede encontrarse en su propia cuenta de AWS o en otra cuenta de AWS. Para obtener más información sobre los autorizadores de Lambda, consulte Uso de autorizadores Lambda de API Gateway.

    > **Note**
    >
    > En la consola de API Gateway, la configuración de `CUSTOM` solo es visible después de que se haya configurado una función de autorizador como se describe en the section called “Configuración de un autorizador de Lambda (consola)”.

> **Important**
>
> El valor de Authorization (Autorización) se aplica a toda la API, no solo a la ruta `$connect`. La ruta `$connect` protege a las demás rutas, porque se la llama en cada conexión.

- Clave de API obligatoria: si lo desea, puede exigir el uso de una clave de API para la ruta `$connect` de una API. Puede utilizar las claves de API junto con los planes de uso para controlar y realizar un seguimiento del acceso a sus API. Para obtener más información, consulte the section called “Planes de uso”.

Configuración de la solicitud de ruta **$connect** con la consola API Gateway

Para configurar la solicitud de la ruta `$connect` para una API de WebSocket mediante la consola de API Gateway:

1. Inicie sesión en la consola de API Gateway, elija la API y, a continuación, elija Routes (Rutas).
2. En Rutas, elija `$connect` o cree una ruta `$connect` según the section called “Creación de una ruta mediante la consola de API Gateway”.
3. En la sección Configuración de la solicitud de ruta, elija Editar.
4. En Autorización, seleccione un tipo de autorización.
5. Para solicitar una API para la ruta `$connect`, seleccione Solicitar clave de API.
6. Seleccione Save changes (Guardar cambios).

Configuración de respuestas de ruta para las API de WebSocket en API Gateway

Las rutas de WebSocket se pueden configurar para la comunicación unidireccional o bidireccional. API Gateway no pasará la respuesta del backend a través de la respuesta de la ruta, a menos configure una respuesta de ruta.

> **Note**
>
> Solo puede definir la respuesta de la ruta `$default` para las API de WebSocket. Puede utilizar una respuesta de integración para manipular la respuesta de un servicio de backend. Para obtener más información, consulte the section called “Información general sobre las respuestas de integración”.

Puede configurar respuestas de ruta y expresiones de selección de respuestas mediante la consola de API Gateway o la AWS CLI o un AWS SDK.

Para obtener más información sobre las expresiones de selección de respuesta de ruta, consulte the section called “”.

Temas

- Configurar una respuesta de ruta mediante la consola de API Gateway
- Configuración de una respuesta de ruta con la AWS CLI

Configurar una respuesta de ruta mediante la consola de API Gateway

Tras crear una API de WebSocket y asociar una función de Lambda proxy a la ruta predeterminada, puede configurar la respuesta de la ruta mediante la consola de API Gateway:

1. Inicie sesión en la consola de API Gateway y elija una API de WebSocket con una integración de función de Lambda de proxy en la ruta `$default`.
2. En Routes (Rutas), elija la ruta `$default`.
3. Elija Habilitar la comunicación bidireccional.
4. Elija Implementar API.
5. Implemente su API en una etapa.

Use el siguiente comando wscat para conectarse a la API. Para obtener más información acerca de `wscat`, consulte the section called “Utilice `wscat` para conectarse y enviar mensajes a una API de WebSocket”.

```
wscat -c wss://api-id.execute-api.us-east-2.amazonaws.com/test
```

Pulse el botón Enter para llamar a la ruta predeterminada. El cuerpo de la función de Lambda debería regresar.

Configuración de una respuesta de ruta con la AWS CLI

El siguiente comando create-route-response permite crear una respuesta de ruta para la ruta `$default`. Puede identificar el ID de la API y el ID de ruta con los comandos get-apis y get-routes.

```
aws apigatewayv2 create-route-response \
    --api-id aabbccddee \
    --route-id 1122334  \
    --route-response-key '$default'
```

El resultado será similar al siguiente:

```
{
    "RouteResponseId": "abcdef",
    "RouteResponseKey": "$default"
}
```

Configuración de una ruta **`$connect`** que requiere un subprotocolo WebSocket

Los clientes pueden usar el campo `Sec-WebSocket-Protocol` para solicitar un subprotocolo WebSocket durante la conexión a su API de WebSocket. Puede configurar una integración de la ruta `$connect` para permitir conexiones solo si un cliente solicita un subprotocolo compatible con su API.

La siguiente función de Lambda de ejemplo devuelve el encabezado `Sec-WebSocket-Protocol` a los clientes. La función establece una conexión a su API solo si el cliente especifica el subprotocolo `myprotocol`.

Para obtener una plantilla de CloudFormation que cree este ejemplo de integración de API y proxy de Lambda, consulte [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;
}
```

Puede usar `wscat` para probar si su API permite conexiones solo si un cliente solicita un subprotocolo compatible con su API. Los siguientes comandos utilizan el indicador `-s` para especificar subprotocolos durante la conexión.

El siguiente comando intenta una conexión con un subprotocolo no compatible. Dado que el cliente especificó el subprotocolo `chat1`, la integración de Lambda devuelve un error 400 y la conexión no es correcta.

```
wscat -c wss://api-id.execute-api.region.amazonaws.com/beta -s chat1
error: Unexpected server response: 400
```

El siguiente comando incluye un subprotocolo admitido en la solicitud de conexión. La integración de Lambda permite la conexión.

```
wscat -c wss://api-id.execute-api.region.amazonaws.com/beta -s chat1,myprotocol
connected (press CTRL+C to quit)
```

Para obtener más información acerca de cómo invocar la API de WebSocket, consulte Invocación de las API de WebSocket.

Control y administración del acceso a las API de WebSocket en API Gateway

API Gateway ofrece diversos mecanismos para controlar y administrar el acceso a su API de WebSocket.

Puede utilizar los siguientes mecanismos para realizar la autenticación y autorización:

- Los roles y las políticas estándar de AWS IAM ofrecen controles de acceso flexibles y robustos. Puede usar roles y políticas de IAM para controlar quién puede crear y administrar sus API, así como quién puede invocarlas. Para obtener más información, consulte Control del acceso a las API de WebSocket con autorización de IAM.
- Las etiquetas de IAM se pueden utilizar con políticas de IAM para controlar el acceso. Para obtener más información, consulte Uso de etiquetas para controlar el acceso a los recursos API de REST de API Gateway.
- Los autorizadores de Lambda son funciones Lambda que controlan el acceso a las API. Para obtener más información, consulte Control del acceso a las API de WebSocket con autorizadores REQUEST de AWS Lambda.

Para aumentar la seguridad, le recomendamos que configure un autorizador para la ruta de `$connect` en todas las API de WebSocket. Es probable que deba hacerlo para asegurar el cumplimiento con diversos marcos normativos. Para obtener más información, consulte Controles de Amazon API Gateway en la Guía del usuario de AWS Security Hub.

Temas

- Control del acceso a las API de WebSocket con autorización de IAM
- Control del acceso a las API de WebSocket con autorizadores REQUEST de AWS Lambda

Control del acceso a las API de WebSocket con autorización de IAM

La autorización de IAM en las API de WebSocket es similar a la de las API REST, con las siguientes excepciones:

- La acción `execute-api` admite `ManageConnections` además de las acciones existentes (`Invoke`, `InvalidateCache`). `ManageConnections` controla el acceso a la API @connections.
- Las rutas de WebSocket utilizan un formato de ARN distinto:

```
arn:aws:execute-api:region:account-id:api-id/stage-name/route-key
```

- La API `@connections` utiliza el mismo formato de ARN que las API de REST:

```
arn:aws:execute-api:region:account-id:api-id/stage-name/POST/@connections
```

> **Important**
>
> Cuando utilice la Autorización de IAM, deberá firmar las solicitudes con Signature Version 4 (SigV4).

Por ejemplo, podría configurar la siguiente política en el cliente. Este ejemplo permite a cualquier usuario enviar un mensaje (`Invoke`) a todas las rutas excepto a una ruta secreta en la etapa `prod` e impide que se manden mensajes a los clientes conectados (`ManageConnections`) en todas las etapas.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "execute-api:Invoke"
            ],
            "Resource": [
                "arn:aws:execute-api:us-east-1:111122223333:api-id/prod/*"
            ]
        },
        {
            "Effect": "Deny",
            "Action": [
                "execute-api:Invoke"
            ],
            "Resource": [
                "arn:aws:execute-api:us-east-1:111122223333:api-id/prod/secret"
            ]
        },
        {
            "Effect": "Deny",
            "Action": [
                "execute-api:ManageConnections"
            ],
            "Resource": [
                "arn:aws:execute-api:us-east-1:111122223333:api-id/*"
            ]
        }
    ]
}
```

Control del acceso a las API de WebSocket con autorizadores REQUEST de AWS Lambda

La función de autorizador de Lambda en las API de WebSocket es similar a la de las API de REST, con las siguientes excepciones:

- Solo puede utilizar una función de autorizador de Lambda para la ruta `$connect`.
- No se pueden utilizar variables de ruta (`event.pathParameters`), ya que la ruta es fija.
- `event.methodArn` es diferente de su equivalente en la API de REST, ya que no tiene ningún método HTTP. En el caso de `$connect`, `methodArn` termina por `"$connect"`:

```
arn:aws:execute-api:region:account-id:api-id/stage-name/$connect
```

- Las variables de contexto de `event.requestContext` son diferentes de las de las API de REST.

En el siguiente ejemplo se muestra una entrada a un autorizador `REQUEST` para una API de 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"
    }
}
```

El siguiente ejemplo de función de autorizador de Lambda es una versión de WebSocket de la función de autorizador de Lambda para las API de REST de la sección the section called “Ejemplos adicionales de funciones del autorizador de Lambda”:

Node.js

```
  // A simple REQUEST authorizer example to demonstrate how to use request
  // parameters to allow or deny a request. In this example, a request is
 parameter
  // in the request context match the specified values of
  // of 'headerValue1' and 'queryValue1' respectively.
           export const handler = function(event, context, callback) {
   console.log('Received event:', JSON.stringify(event, null, 2));

  // Retrieve request parameters from the Lambda function input:
  var headers = event.headers;
  var queryStringParameters = event.queryStringParameters;
  var stageVariables = event.stageVariables;
  var requestContext = event.requestContext;

  // Parse the input for the parameter values
  var tmp = event.methodArn.split(':');
  var apiGatewayArnTmp = tmp[5].split('/');
  var awsAccountId = tmp[4];
  var region = tmp[3];
  var ApiId = apiGatewayArnTmp[0];
  var stage = apiGatewayArnTmp[1];
  var route = apiGatewayArnTmp[2];

  // Perform authorization to return the Allow policy for correct parameters and
  // the 'Unauthorized' error, otherwise.
```

```
    var authResponse = {};
    var condition = {};
     condition.IpAddress = {};

    if (headers.HeaderAuth1 === "headerValue1"
        && queryStringParameters.QueryString1 === "queryValue1") {
         callback(null, generateAllow('me', event.methodArn));
     }  else {
         callback(null, generateDeny('me', event.methodArn));
     }
}

// Helper function to generate an IAM policy
var generatePolicy = function(principalId, effect, resource) {
    // Required output:
    var authResponse = {};
     authResponse.principalId = principalId;
    if (effect && resource) {
        var policyDocument = {};
         policyDocument.Version = '2012-10-17       '; // default version
        policyDocument.Statement = [];
        var statementOne = {};
         statementOne.Action = 'execute-api:Invoke'; // default action
        statementOne.Effect = effect;
         statementOne.Resource = resource;
         policyDocument.Statement[0] = statementOne;
         authResponse.policyDocument = policyDocument;
     }
    // Optional output with custom properties of the String, Number or Boolean type.
    authResponse.context = {
        "stringKey": "stringval",
        "numberKey": 123,
        "booleanKey": true
     };
    return authResponse;
}

var generateAllow = function(principalId, resource) {
    return generatePolicy(principalId, 'Allow', resource);
}

var generateDeny = function(principalId, resource) {
    return generatePolicy(principalId, 'Deny', resource);
```

```
}
```

Python

```
# A simple REQUEST authorizer example to demonstrate how to use request
# parameters to allow or deny a request. In this example, a request is
# authorized if the client-supplied HeaderAuth1 header and QueryString1 query
 parameter
# in the request context match the specified values of
# of 'headerValue1' and 'queryValue1' respectively.

import json


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

    # Retrieve request parameters from the Lambda function input:
    headers = event['headers']
    queryStringParameters = event['queryStringParameters']
    stageVariables = event['stageVariables']
    requestContext = event['requestContext']

    # Parse the input for the parameter values
    tmp = event['methodArn'].split(':')
    apiGatewayArnTmp = tmp[5].split('/')
    awsAccountId = tmp[4]
    region = tmp[3]
    ApiId = apiGatewayArnTmp[0]
    stage = apiGatewayArnTmp[1]
    route = apiGatewayArnTmp[2]

    # Perform authorization to return the Allow policy for correct parameters
    # and the 'Unauthorized' error, otherwise.

    authResponse = {}
    condition = {}
    condition['IpAddress'] = {}

    if (headers['HeaderAuth1'] ==
            "headerValue1" and queryStringParameters["QueryString1"] ==
 "queryValue1"):
        response = generateAllow('me', event['methodArn'])
```

```
        print('authorized')
        return json.loads(response)
    else:
        response = generateDeny('me', event['methodArn'])
        print('unauthorized')
        return json.loads(response)

    # Help function to generate IAM policy


def generatePolicy(principalId, effect, resource):
    authResponse = {}
    authResponse['principalId'] = principalId
    if (effect and resource):
        policyDocument = {}
        policyDocument['Version'] = '2012-10-17       '
        policyDocument['Statement'] = []
        statementOne = {}
        statementOne['Action'] = 'execute-api:Invoke'
        statementOne['Effect'] = effect
        statementOne['Resource'] = resource
        policyDocument['Statement'] = [statementOne]
        authResponse['policyDocument'] = policyDocument

    authResponse['context'] = {
        "stringKey": "stringval",
        "numberKey": 123,
        "booleanKey": True
    }

    authResponse_JSON = json.dumps(authResponse)

    return authResponse_JSON


def generateAllow(principalId, resource):
    return generatePolicy(principalId, 'Allow', resource)


def generateDeny(principalId, resource):
    return generatePolicy(principalId, 'Deny', resource)
```

Para configurar la función de Lambda anterior como una función de autorizador `REQUEST` para una API de WebSocket, siga el mismo procedimiento que para las API de REST.

Para configurar la ruta `$connect` de forma que utilice este autorizador de Lambda en la consola, seleccione o cree la ruta `$connect`. En la sección Configuración, elija Editar. Seleccione su autorizador en el menú desplegable Autorización y, a continuación, elija Guardar cambios.

Para probar el autorizador, tiene que crear una conexión nueva. El cambio de autorizador en `$connect` no afecta a los clientes que ya están conectados. Al conectarse a la API de WebSocket, tiene que proporcionar valores para todas las fuentes de identidad configuradas. Por ejemplo, puede conectarse enviando una cadena de consulta y un encabezado válidos mediante `wscat`, como se muestra en el siguiente ejemplo:

```
wscat -c 'wss://myapi.execute-api.us-east-1.amazonaws.com/beta?
QueryString1=queryValue1' -H HeaderAuth1:headerValue1
```

Si intenta conectarse sin un valor de identidad válido, recibirá una respuesta `401`:

```
wscat -c wss://myapi.execute-api.us-east-1.amazonaws.com/beta
error: Unexpected server response: 401
```

Integraciones de API de WebSocket en API Gateway

Una vez configurada una ruta de API, debe integrarlo con un punto de conexión en el backend. El punto de conexión del backend también se conoce como punto de conexión de integración y puede ser una función de Lambda, un punto de conexión HTTP o una acción del servicio de AWS. a integración de la API tiene una solicitud de integración y una respuesta de integración.

En esta sección, puede aprender a configurar solicitudes de integración y respuestas de integración para su API de WebSocket.

Temas

- Configuración de una solicitud de integración de API de WebSocket en API Gateway
- Configuración de una respuesta de integración de API de WebSocket en API Gateway

Configuración de una solicitud de integración de API de WebSocket en API Gateway

La configuración de una solicitud de integración implica lo siguiente:

- Elegir una clave de ruta para integrarla en el backend.
- Especificación del punto de conexión de backend que se va a invocar. Las API de WebSocket admiten los siguientes tipos de integración:
  - `AWS_PROXY`
  - `AWS`
  - `HTTP_PROXY`
  - `HTTP`
  - `MOCK`

  Para obtener más información sobre los tipos de integración, consulte IntegrationType en la API de REST de API Gateway V2.
- Configurar cómo transformar los datos de la solicitud de ruta, si fuera necesario, en datos de solicitud de integración mediante la especificación de una o varias plantillas de solicitud.

Configuración de una solicitud de integración de la API de WebSocket mediante la consola de API Gateway

Para agregar una solicitud de integración a una ruta en una API de WebSocket mediante la consola de API Gateway

1. Inicie sesión en la consola de API Gateway, elija la API y, a continuación, elija Routes (Rutas).
2. En Routes (Rutas), elija la ruta.
3. Elija la pestaña Solicitud de integración y, a continuación, en la sección Configuración de solicitud de integración, elija Editar.
4. En Tipo de integración, seleccione una de las siguientes opciones:

   - Elija Función de Lambda solo si la API se va a integrar con una función de AWS Lambda que ya ha creado en esta cuenta o en otra.

     Para crear una nueva función de Lambda en AWS Lambda, establecer un permiso a nivel de recursos en la función de Lambda o realizar cualquier otra acción de servicio Lambda, elija Servicio de AWS en su lugar.
   - Elija HTTP si la API se va a integrar con un punto de conexión HTTP existente. Para obtener más información, consulte Integraciones de HTTP para las API de REST en API Gateway.

- Elija Mock (Simulación) si desea generar respuestas de la API directamente a partir de API Gateway, sin necesidad de un backend de integración. Para obtener más información, consulte Simulación de integraciones para las API de REST en API Gateway.
- Elija Servicio de AWS si la API se va a integrar directamente con un servicio de AWS.
- Elija Enlace de VPC si la API va a utilizar un `VpcLink` como un punto de conexión de integración privada. Para obtener más información, consulte Configuración de una integración privada.

5. Si elige Función de Lambda, proceda de la forma siguiente:

   a. En Usar la integración de proxy Lambda, elija la casilla si piensa usar la Integración de proxy Lambda o la Integración de proxy Lambda entre cuentas.

   b. En Función de Lambda, especifique la función de una de las siguientes formas:

      - Si la función de Lambda está en la misma cuenta, ingrese el nombre de la función y, a continuación, seleccione la función de la lista desplegable.

        > **Note**
        >
        > Si lo desea, el nombre de la función puede incluir su alias o especificación de versión, como en `HelloWorld`, `HelloWorld:1` o `HelloWorld:alpha`.

      - Si la función se encuentra en una cuenta diferente, escriba el ARN de la función.

   c. Si desea utilizar el valor predeterminado del tiempo de espera, que es de 29 segundos, mantenga activado el Tiempo de espera predeterminado. Para establecer un tiempo de espera personalizado, elija Tiempo de espera predeterminado e ingrese un valor de tiempo de espera comprendido entre `50` y `29000` milisegundos.

6. Si ha elegido HTTP, siga las instrucciones del paso 4 que figuran en the section called “Configuración de una solicitud de integración mediante la consola”.
7. Si ha elegido Mock (Simulación), continúe con el paso Request Templates (Plantillas de solicitud).
8. Si elige Servicio de AWS, siga las instrucciones del paso 6 que figuran en the section called “Configuración de una solicitud de integración mediante la consola”.
9. Si elige Enlace de VPC, haga lo siguiente:

   a. En Integración de proxy VPC, elija la casilla si desea que sus solicitudes se transfieran por proxy al punto de conexión del `VPCLink`.

b. En HTTP method (Método HTTP), elija el tipo de método HTTP que más se parezca al método del backend HTTP.

c. En la lista desplegable Enlace de VPC, seleccione un enlace de VPC. Puede seleccionar `[Use Stage Variables]` e ingresar **`${stageVariables.vpcLinkId}`** en el cuadro de texto situado debajo de la lista.

   Puede definir la variable de etapa `vpcLinkId` después de implementar la API en una etapa y establecer su valor en el ID del `VpcLink`.

d. En URL del punto de conexión, escriba la dirección URL del backend HTTP que desea que utilice esta integración.

e. Si desea utilizar el valor predeterminado del tiempo de espera, que es de 29 segundos, mantenga activado el Tiempo de espera predeterminado. Para establecer un tiempo de espera personalizado, elija Tiempo de espera predeterminado e ingrese un valor de tiempo de espera comprendido entre `50` y `29000` milisegundos.

10. Seleccione Save changes (Guardar cambios).

11. En Plantillas de solicitud, haga lo siguiente:

    a. Para ingresar una Expresión de selección de plantillas, en Plantillas de solicitud, elija Editar.

    b. Ingrese una Expresión de selección de plantillas. Use una expresión que API Gateway busque en la carga del mensaje. Si la encuentra, se evalúa y el resultado es un valor de clave de plantilla que se utiliza para seleccionar la plantilla de asignación de datos que debe aplicarse a los datos de la carga del mensaje. En el siguiente paso se crea la plantilla de mapeo de datos. Elija Editar para guardar los cambios.

    c. Elija Crear plantilla para crear la plantilla de mapeo de datos. En Clave de plantilla, ingrese el valor de clave de plantilla que se utiliza para seleccionar la plantilla de mapeo de datos que debe aplicarse a los datos de la carga del mensaje. A continuación, ingrese una plantilla de mapeo. Seleccione Crear plantilla.

       Para obtener información sobre las expresiones de selección de plantillas, consulte the section called “Expresiones de selección de plantilla”.

Configurar una solicitud de integración mediante la AWS CLI

Puede utilizar la AWS CLI para configurar una solicitud de integración para una ruta en una API de WebSocket tal y como se muestra en el siguiente ejemplo, que crea una integración simulada:

1. Cree un archivo denominado `integration-params.json` con el siguiente contenido:

```
{"PassthroughBehavior": "WHEN_NO_MATCH", "TimeoutInMillis": 29000,
 "ConnectionType": "INTERNET", "RequestTemplates": {"application/json":
 "{\"statusCode\":200}"}, "IntegrationType": "MOCK"}
```

2. El siguiente comando create-integration permite crear la integración simulada.

```
aws apigatewayv2 --region us-east-1 create-integration --api-id aabbccddee --cli-
input-json file://integration-params.json
```

   El resultado será similar al siguiente:

```
{
    "PassthroughBehavior": "WHEN_NO_MATCH",
    "TimeoutInMillis": 29000,
    "ConnectionType": "INTERNET",
    "IntegrationResponseSelectionExpression": "${response.statuscode}",
    "RequestTemplates": {
        "application/json": "{\"statusCode\":200}"
    },
    "IntegrationId": "0abcdef",
    "IntegrationType": "MOCK"
}
```

También puede configurar una solicitud de integración para una integración de proxy mediante la AWS CLI.

1. Cree una función de Lambda en la consola de Lambda y asígnela un rol de ejecución de funciones Lambda básico.
2. Use el siguiente comando create-integration para crear la integración.

```
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
```

El resultado será similar al siguiente:

```
{
    "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"
}
```

Formato de entrada de una función de Lambda para la integración de proxy para API de WebSocket

Con la integración de proxy de Lambda, API Gateway asigna toda la solicitud de cliente al parámetro de entrada `event` de la función de Lambda del backend. En el siguiente ejemplo se muestra la estructura del evento de entrada de la ruta `$connect` y el evento de entrada de la ruta `$disconnect` que API Gateway envía a una integración de proxy de Lambda.

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
}
```

Configuración de una respuesta de integración de API de WebSocket en API Gateway

En la siguiente sección se proporciona una breve información general de las respuestas de integración para la API de WebSocket y cómo configurar una respuesta de integración para una API de WebSocket.

Temas

- Información general sobre las respuestas de integración
- Respuestas de integración para la comunicación bidireccional
- Configuración de una respuesta de integración mediante la consola de API Gateway
- Configurar una respuesta de integración mediante la AWS CLI

Información general sobre las respuestas de integración

La respuesta de integración de API Gateway es una manera de modelar y manipular la respuesta desde un servicio de backend. Existen algunas diferencias en la configuración de una API de REST frente a la de una respuesta de integración de API de WebSocket, pero conceptualmente el comportamiento es el mismo.

Las rutas de WebSocket se pueden configurar para la comunicación unidireccional o bidireccional.

- Cuando una ruta se configura para la comunicación bidireccional, una respuesta de integración le permite configurar transformaciones de la carga del mensaje que se devuelve, de forma similar a las respuestas de integración para las API de REST.
- Si una ruta se configura para la comunicación unidireccional, independientemente de la configuración de respuesta de integración, no se devolverá ninguna respuesta a través del canal de WebSocket una vez procesado el mensaje.

API Gateway no pasará la respuesta del backend a través de la respuesta de la ruta, a menos configure una respuesta de ruta. Para obtener más información sobre cómo configurar una respuesta de ruta, consulte the section called “Configuración de respuestas de ruta de API de WebSocket”.

Respuestas de integración para la comunicación bidireccional

Las integraciones se pueden dividir en integraciones de proxy e integraciones que no son de proxy.

> **Important**
>
> En las integraciones de proxy, API Gateway pasa automáticamente la salida del backend al intermediario como la carga completa. No hay ninguna respuesta de integración.

En las integraciones que no son de proxy, es necesario configurar al menos una respuesta de integración:

- Lo ideal sería que una de las respuestas de integración actuase como método catch-all cuando no se puede realizar ninguna elección explícita. Este caso predeterminado se representa mediante la configuración de la clave de respuesta de integración `$default`.
- En el resto de los casos, la clave de respuesta de integración funciona como una expresión regular. Debe seguir el formato `"/expression/"`.

Para las integraciones HTTP que no sean de proxy:

- API Gateway intentará emparejar el código de estado HTTP de la respuesta del backend. La clave de respuesta de integración funcionará como una expresión regular en este caso. Si no se encuentra una coincidencia, se elige `$default` como respuesta de integración.
- La expresión de selección de plantillas, tal y como se describe anteriormente, funciona de forma idéntica. Por ejemplo:
  - `/2\d\d/`: recibe y transforma respuestas correctas
  - `/4\d\d/`: recibe y transforma errores de solicitud incorrecta
  - `$default`: recibe y transforma todas las respuestas inesperadas

Para obtener más información sobre las expresiones de selección de plantillas, consulte the section called “Expresiones de selección de plantilla”.

Configuración de una respuesta de integración mediante la consola de API Gateway

Para configurar una respuesta de integración de ruta en una API de WebSocket mediante la consola de API Gateway:

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Elija la API de WebSocket y elija la ruta.
3. Elija la pestaña Solicitud de integración y, a continuación, en la sección Configuración de solicitud de integración, elija Crear respuesta de integración.
4. En Clave de respuesta, introduzca un valor que se encontrará en la clave de respuesta que aparece en el mensaje saliente tras evaluar la expresión de selección de respuesta. Por ejemplo, puede ingresar **`/4\d\d/`** para recibir y transformar los errores de solicitudes erróneas o ingresar **`$default`** para recibir y transformar todas las respuestas que coincidan con la expresión de selección de plantillas.
5. En Expresión de selección de plantillas, ingrese una expresión de selección para evaluar el mensaje saliente.
6. Elija Crear respuesta.
7. También puede definir una plantilla de mapeo para configurar las transformaciones de la carga útil de los mensajes devueltos. Seleccione Crear plantilla.
8. Ingrese un nombre de clave. Si elige la expresión de selección de plantillas predeterminada, ingrese **`\$default`**.
9. En Plantilla de respuesta, ingrese la plantilla de mapeo en el editor de código.
10. Seleccione Crear plantilla.
11. Elige Implementar API para implementar su API.

Use el siguiente comando wscat para conectarse a la API. Para obtener más información acerca de `wscat`, consulte the section called “Utilice `wscat` para conectarse y enviar mensajes a una API de WebSocket”.

```
wscat -c wss://api-id.execute-api.us-east-2.amazonaws.com/test
```

Cuando llame a la ruta, la carga útil del mensaje devuelto debería regresar.

Configurar una respuesta de integración mediante la AWS CLI

El siguiente comando create-integration-response permite crear una respuesta de integración `$default`:

```
aws apigatewayv2 create-integration-response \
    --api-id vaz7da96z6 \
    --integration-id a1b2c3 \
    --integration-response-key '$default'
```

Solicitud de la validación de las API de WebSocket en API Gateway

Puede configurar API Gateway para que realice la validación en una solicitud de ruta antes de continuar con la solicitud de integración. Si la validación falla, API Gateway falla la solicitud sin llamar a su back-end, envía una respuesta de puerta de enlace "Cuerpo de solicitud incorrecto" al cliente y publica los resultados de validación en CloudWatch Logs. El uso de la validación de esta manera reduce las llamadas innecesarias a su back-end de API.

Expresiones de selección de modelo

Puede utilizar una expresión de selección de modelo para validar dinámicamente solicitudes dentro de la misma ruta. La validación del modelo se produce si proporciona una expresión de selección de modelo para integraciones proxy o no proxy. Es posible que necesite definir el modelo de `$default` como una alternativa cuando no se encuentre ningún modelo coincidente. Si no hay ningún modelo coincidente y `$default` no está definido, la validación falla. La expresión de selección tiene el aspecto `Route.ModelSelectionExpression` y se evalúa como la clave para `Route.RequestModels`.

Al definir una ruta para una API de WebSocket, puede especificar si lo desea una expresión de selección de modelo. Esta expresión se evalúa para seleccionar el modelo que se utilizará para la validación del cuerpo cuando se reciba una solicitud. La expresión se evalúa en una de las entradas de la propiedad de una ruta `requestmodels`.

Un modelo se expresa como un esquema JSON y describe la estructura de datos del cuerpo de la solicitud. La naturaleza de estas expresiones de selección le permite elegir de forma dinámica el modelo con el que debe realizarse la validación en tiempo de ejecución para una ruta determinada. Para obtener información sobre cómo crear un modelo, consulte the section called “Modelos de datos para las API de REST”.

Configuración de la validación de solicitudes mediante la consola de API Gateway

En el siguiente ejemplo se muestra cómo configurar la validación de la solicitud en una ruta.

En primer lugar, debe crear un modelo y, a continuación, crear una ruta. A continuación, debe configurar la validación de la solicitud en la ruta que acaba de crear. Por último, debe implementar y probar la API. Para completar este tutorial, necesita una API de WebSocket con `$request.body.action` como expresión de selección de rutas y un punto de conexión de integración para la nueva ruta.

También es necesario `wscat` para conectarse a la API. Para obtener más información, consulte the section called “Utilice `wscat` para conectarse y enviar mensajes a una API de WebSocket”.

Para crear un modelo

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Elija una API de WebSocket.
3. En el panel de navegación principal, elija Modelos.
4. Seleccione Crear modelo.
5. En Nombre, escriba **`emailModel`**.
6. En Tipo de contenido, ingrese **`application/json`**.
7. En Esquema del modelo, escriba el siguiente modelo:

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "type" : "object",
    "required" : [ "address"],
    "properties" : {
        "address": {
            "type": "string"
        }
    }
}
```

   Este modelo requiere que la solicitud contenga una dirección de correo electrónico.
8. Seleccione Save.

En este paso, se crea una ruta para la API de WebSocket.

Para crear una ruta

1. En el panel de navegación principal, elija Rutas.
2. Elija Create route (Crear ruta).
3. Para Route key (Clave de ruta), ingrese **`sendMessage`**.
4. Elija un tipo de integración y especifique un punto de conexión de integración. Para obtener más información consulte () the section called “Integraciones”.
5. Elija Create route (Crear ruta).

En este paso, debe configurar la validación de solicitudes para la ruta `sendMessage`.

Configuración de la validación de solicitudes

1. En la pestaña Solicitud de ruta, en Configuración de solicitud de ruta, elija Editar.
2. Para Expresión de selección de modelo, ingrese **`${request.body.messageType}`**.

   API Gateway usa la propiedad `messageType` para validar la solicitud entrante.
3. Elija Agregar modelo de solicitud.
4. Para Clave de modelo, ingrese **`email`**.
5. Para Modelo, elija emailModel.

   API Gateway valida los mensajes entrantes con la propiedad `messageType` establecida en `email` frente a este modelo.

   > **Note**
   >
   > Si API Gateway no puede hacer coincidir la expresión de selección del modelo con una clave de modelo, selecciona el modelo `$default`. Si no hay ningún modelo `$default`, la validación produce un error. Para las API de producción, se recomienda crear un modelo `$default`.
6. Seleccione Save changes (Guardar cambios).

En este paso, debe implementar y probar la API.

Implementación y prueba de la API

1. Elija Implementar API.
2. Elija la etapa que desee en el menú desplegable o escriba el nombre de una etapa nueva.
3. Elija Implementar.
4. En el panel de navegación principal, elija Etapas.
5. Copie la URL de WebSocket de la API. La dirección URL debe tener un aspecto similar al siguiente: `wss://abcdef123.execute-api.us-east-2.amazonaws.com/production`.
6. Abra un nuevo terminal y ejecute el comando wscat con los siguientes parámetros.

```
wscat -c wss://abcdef123.execute-api.us-west-2.amazonaws.com/production
```

```
Connected (press CTRL+C to quit)
```

7. Use el siguiente comando para probar la API.

```
{"action": "sendMessage", "messageType": "email"}
```

```
{"message": "Invalid request body", "connectionId":"ABCD1=234",
 "requestId":"EFGH="}
```

API Gateway producirá un error en la solicitud.

Use el siguiente comando para enviar una solicitud válida a la API.

```
{"action": "sendMessage", "messageType": "email", "address":
 "mary_major@example.com"}
```

Transformaciones de datos para las API de WebSocket en API Gateway

En API Gateway, una solicitud de método de API de WebSocket puede aceptar una carga en un formato diferente del de la carga de la solicitud de integración, según requiera el backend. Del mismo modo, el backend puede devolver una carga de respuesta de integración diferente de la carga de respuesta del método que espera el frontend.

API Gateway le permite utilizar transformaciones de plantillas de asignación para asignar la carga útil de una solicitud de método a la solicitud de integración correspondiente y de una respuesta de integración a la respuesta de método correspondiente. Cree una plantilla de asignación y especifique una expresión de selección de plantilla para determinar qué plantilla se utilizará para realizar las transformaciones de datos necesarias.

Puede utilizar asignaciones de datos para asignar datos de una solicitud de ruta a una integración con backend. Para obtener más información, consulte the section called “Asignación de datos”.

Plantillas y modelos de asignación

Una plantilla de asignación es un script expresado en Velocity Template Language (VTL) que se aplica a la carga mediante expresiones JSONPath. Para obtener más información acerca de las plantillas de mapeo de API Gateway, consulte Asignación de las transformaciones de plantillas de asignación para las API de REST en API Gateway.

La carga puede tener un modelo de datos de acuerdo con el esquema JSON, borrador 4. No tiene que definir un modelo para crear una plantilla de asignación. Sin embargo, un modelo puede ayudarle a crear una plantilla porque API Gateway genera un esquema de plantilla basado en un modelo proporcionado. Para obtener más información acerca de los modelos de API Gateway, consulte Modelos de datos para las API de REST.

Expresiones de selección de plantilla

Para transformar una carga con una plantilla de mapeo, especifique una expresión de selección de plantilla de API de WebSocket en una solicitud de integración o respuesta de integración. Esta expresión se evalúa para determinar la plantilla de entrada o de salida (si procede) que se va a utilizar para transformar ya sea el cuerpo de la solicitud en el cuerpo de la solicitud de integración (a través de una plantilla de entrada) o el cuerpo de la respuesta de integración en el cuerpo de la respuesta de ruta (a través de una plantilla de salida).

`Integration.TemplateSelectionExpression` admite `${request.body.jsonPath}` y valores estáticos.

`IntegrationResponse.TemplateSelectionExpression` admite `${request.body.jsonPath}`, `${integration.response.statuscode}`, `${integration.response.header.headerName}`, `${integration.response.multivalueheader.headerName}` y valores estáticos.

Expresiones de selección de respuesta de integración

Al configurar una respuesta de integración para una API de WebSocket, puede especificar si lo desea una expresión de selección de respuesta de integración. Esta expresión determina qué valor de `IntegrationResponse` debe seleccionarse cuando regresa una integración. El valor de esta expresión se encuentra actualmente restringido por API Gateway, tal y como se define a continuación. Tenga en cuenta que esta expresión solo se aplica a las integraciones que no sean de proxy; una integración de proxy simplemente transfiere la carga de respuesta de vuelta al intermediario sin realizar ningún modelado ni modificación.

A diferencia del resto de expresiones de selección precedentes, actualmente esta expresión admite un formato de coincidencia de patrones. La expresión debe encerrarse entre barras inclinadas.

Actualmente, el valor es fijo en función de la propiedad `integrationType`:

- Para integraciones basadas en Lambda, es `$integration.response.body.errorMessage`.
- En las integraciones `HTTP` y `MOCK`, es `$integration.response.statuscode`.
- En `HTTP_PROXY` y `AWS_PROXY`, la expresión no se utiliza, ya que se está solicitando que la carga pase hacia el intermediario.

Configuración de asignación de datos para las API de WebSocket en API Gateway

La asignación de datos le permite asignar datos de una solicitud de ruta a una integración del backend.

> **Note**
>
> El mapeo de datos para las API de WebSocket no se admite en Consola de administración de AWS. Debe utilizar la AWS CLI, AWS CloudFormation o un SDK para configurar la asignación de datos.

Temas

- Asignar datos de solicitud de ruta a parámetros de solicitud de integración
- Ejemplos

Asignar datos de solicitud de ruta a parámetros de solicitud de integración

Los parámetros de solicitud de integración se pueden asignar desde cualquier parámetro de solicitud de ruta definido, el cuerpo de la solicitud, variables `context` o `stage` y valores estáticos.

En la siguiente tabla se muestran las expresiones de asignación de datos de las solicitudes de integración. En la tabla, *PARAM_NAME* es el nombre de un parámetro de solicitud de ruta del tipo de parámetro especificado. Debe coincidir con la expresión regular `'^[a-zA-Z0-9._$-]+$]'`. *JSONPath_expression* es una expresión JSONPath para un campo JSON del cuerpo de la solicitud.

Origen de datos asignado	Expresión de asignación
Cadena de consulta de solicitud (compatible solo con la ruta `$connect`)	`route.request.querystring.` *PARAM_NAME*
Encabezado de solicitud (compatible solo con la ruta `$connect`)	`route.request.header.` *PARAM_NAME*
Cadena de consulta de solicitud de varios valores (compatible solo con la ruta `$connect`)	`route.request.multivaluequerystring.` *PARAM_NAME*
Encabezado de solicitud de varios valores (solo compatible con la ruta `$connect`)	`route.request.multivalueheader.` *PARAM_NAME*
Cuerpo de la solicitud	`route.request.body.` *JSONPath_EXPRESSION*
Variables de etapa	`stageVariables.` *VARIABLE_NAME*
Variables de contexto	`context.`*VARIABLE_NAME* que debe ser alguna de las variables de contexto admitidas.
Valor estático	*'STATIC_VALUE'* . *STATIC_VALUE* es un literal de cadena que se debe incluir entre comillas simples.

Al crear una asignación de datos, mediante la AWS CLI asegúrese de seguir el formato correcto para usar literales con cadenas en la AWS CLI. Para obtener más información, consulte Uso de

entrecomillado y literales con cadenas en la AWS CLI de la Guía del usuario de AWS Command Line Interface.

Ejemplos

En los siguientes ejemplos de la AWS CLI se configuran asignaciones de datos. Para obtener una plantilla de CloudFormation, consulte `websocket-data-mapping.yaml`.

Asignar el connectionId de un cliente a un encabezado en una solicitud de integración

El siguiente comando update-integration permite asignar el `connectionId` de un cliente a un encabezado `connectionId` en la solicitud a una integración de backend:

```
aws apigatewayv2 update-integration \
    --integration-id abc123 \
    --api-id a1b2c3d4 \
    --request-parameters
 'integration.request.header.connectionId'='context.connectionId'
```

Asignar un parámetro de cadena de consulta a un encabezado en una solicitud de integración

En el siguiente ejemplo, se asigna un parámetro de cadena de consulta `authToken` a un encabezado `authToken` en la solicitud de integración.

1. Utilice el siguiente comando update-route para agregar el parámetro de cadena de consulta `authToken` a los parámetros de solicitud de ruta.

   ```
   aws apigatewayv2 update-route --route-id 0abcdef \
       --api-id a1b2c3d4 \
       --request-parameters '{"route.request.querystring.authToken": {"Required":
    false}}'
   ```

2. Utilice el siguiente comando update-integration para asignar el parámetro de cadena de consulta al encabezado `authToken` en la solicitud a la integración de backend.

   ```
   aws apigatewayv2 update-integration \
       --integration-id abc123 \
       --api-id a1b2c3d4 \
       --request-parameters
    'integration.request.header.authToken'='route.request.querystring.authToken'
   ```

3. (Opcional) Si es necesario, utilice el siguiente comando delete-route-request-parameter para eliminar el parámetro de cadena de consulta `authToken` de los parámetros de solicitud de ruta.

```
aws apigatewayv2 delete-route-request-parameter \
    --route-id 0abcdef \
    --api-id a1b2c3d4 \
    --request-parameter-key 'route.request.querystring.authToken'
```

Referencia de la plantilla de asignación de la API de WebSocket para API Gateway

En esta sección se resume el conjunto de variables compatibles actualmente con las API de WebSocket en API Gateway.

Parámetro	Descripción
`$context.connectionId`	Un ID único para la conexión que puede utilizar para realizar una devolución de llamada al cliente. El ID de conexión es una cadena de longitud variable cuya longitud y formato pueden cambiar.
`$context.connectedAt`	Hora de la conexión en formato de tiempo Unix.
`$context.domainName`	Nombre de dominio para la API de WebSocket . Se puede utilizar para realizar una devolució n de llamada al cliente (en lugar de un valor de código rígido).
`$context.eventType`	El tipo de evento: `CONNECT`, `MESSAGE` o `DISCONNECT` .
`$context.messageId`	Un ID único del lado del servidor para un mensaje. Solo está disponible si `$context.eventType` es `MESSAGE`.
`$context.routeKey`	La clave de ruta seleccionada.

Parámetro	Descripción
`$context.requestId`	Igual que `$context.extendedRequestId` .
`$context.extendedRequestId`	Un ID generado de forma automática para la llamada a la API, que contiene más información útil para la depuración o la resolución de problemas.
`$context.apiId`	El identificador que API Gateway asigna a su API.
`$context.authorizer.principalId`	La identificación de usuario principal asociada con el token enviado por el cliente y devuelto de una función de Lambda del autorizador de Lambda de API Gateway (que anteriormente se denominaba “autorizador personalizado”).

`$context.authorizer.` *property*	El valor en forma de cadena del par clave-valor especificado de la asignación `context` que devuelve la función de Lambda del autorizador de Lambda de API Gateway. Por ejemplo, si el autorizador devuelve la siguiente asignación de `context`:
	`"context" : {`
	`  "key": "value",`
	`  "numKey": 1,`
	`  "boolKey": true`
	`}`
	la llamada a `$context.authorizer.key` devuelve la cadena `"value"`, la llamada a `$context.authorizer.numKey` devuelve la cadena `"1"` y la llamada a `$context.authorizer.boolKey` devuelve la cadena `"true"`.

Parámetro	Descripción
`$context.error.messageString`	El valor entrecomillado de `$context.error.message`, es decir, `"$context.error.message"`.
`$context.error.validationErrorString`	Una cadena que contiene un mensaje de error de validación detallado.
`$context.identity.accountId`	El ID de cuenta de AWS asociado con la solicitud.
`$context.identity.apiKey`	Clave del propietario de API asociada a la solicitud de API habilitada para claves.
`$context.identity.apiKeyId`	ID de clave de API asociado a la solicitud de API habilitada para claves
`$context.identity.caller`	El identificador principal del intermediario que realiza la solicitud.

Parámetro	Descripción
`$context.identity.cognitoAuthenticationProvider`	Una lista separada por comas de todos los proveedores de autenticación de Amazon Cognito utilizados por el intermediario que realiza la solicitud. Solo está disponible si la solicitud se firmó con las credenciales de Amazon Cognito. Por ejemplo, para una identidad de un grupo de usuarios de Amazon Cognito, `cognito-idp.`*`region`*`.amazonaws.com/`*`user_pool_id`* `,cognito-idp.`*`region`*`.amazonaws.com/`*`user_pool_id`* `:CognitoSignIn:`*`token subject claim`* Consulte Uso de las identidades federadas en la Guía para desarrolladores de Amazon Cognito para obtener información sobre los proveedores de autenticación de Amazon Cognito disponibles.
`$context.identity.cognitoAuthenticationType`	El tipo de autenticación de Amazon Cognito del intermediario que realiza la solicitud. Solo está disponible si la solicitud se firmó con las credenciales de Amazon Cognito. Los valores posibles incluyen `authenticated` para identidades autenticadas y `unauthenticated` para identidades no autenticadas.
`$context.identity.cognitoIdentityId`	El ID de identidad de Amazon Cognito del intermediario que realiza la solicitud. Solo está disponible si la solicitud se firmó con las credenciales de Amazon Cognito.

Parámetro	Descripción
`$context.identity.cognitoId entityPoolId`	El ID del grupo de identidades de Amazon Cognito del intermediario que realiza la solicitud. Solo está disponible si la solicitud se firmó con las credenciales de Amazon Cognito.
`$context.identity.sourceIp`	La dirección IP de origen de la conexión TCP inmediata que realiza la solicitud al punto de enlace de API Gateway.
`$context.identity.user`	El identificador principal del usuario que realiza la solicitud.
`$context.identity.userAgent`	El agente de usuario del intermediario de la API.
`$context.identity.userArn`	El Nombre de recurso de Amazon (ARN) del usuario identificado después de la autentica ción.
`$context.requestTime`	Hora de la solicitud en formato CLF-(`dd/MMM/yyyy:HH:mm:ss +-hhmm`).
`$context.requestTimeEpoch`	Hora de la solicitud en formato Epoch en milisegundos.
`$context.stage`	La etapa de implementación de la llamada a la API (por ejemplo, Beta o Prod).
`$context.status`	Estado de la respuesta.
`$input.body`	Devuelve la carga bruta como una cadena.

Parámetro	Descripción
`$input.json(x)`	Esta función evalúa una expresión JSONPath y devuelve los resultados como una cadena JSON. Por ejemplo, `$input.json('$.pets')` devolverá una cadena JSON que representa la estructura de "pets" (mascotas). Para obtener más información acerca de JSONPath, consulte JSONPath o JSONPath for Java.

Parámetro	Descripción
`$input.path(x)`	Toma una cadena de expresión JSONPath (x) y devuelve una representación del resultado en forma de objeto JSON. Esto le permite tener acceso y manipular los elementos de la carga de forma nativa en Apache Velocity Template Language (VTL). Por ejemplo, si la expresión `$input.path('$.pets')` devuelve un objeto como este: `[ { "id": 1, "type": "dog", "price": 249.99 }, { "id": 2, "type": "cat", "price": 124.99 }, { "id": 3, "type": "fish", "price": 0.99 } ]` `$input.path('$.pets').count()` devolvería `"3"`. Para obtener más información acerca de JSONPath, consulte JSONPath o JSONPath for Java.
`$stageVariables.` *`<variable_name>`*	*`<variable_name>`* representa un nombre de variable de etapa.

Parámetro	Descripción
`$stageVariables['` *`<variable_name>`* `']`	*`<variable_name>`* representa cualquier nombre de variable de etapa.
`${stageVariables['` *`<variable_name>`* `']}`	*`<variable_name>`* representa cualquier nombre de variable de etapa.
`$util.escapeJavaScript()`	Aplica caracteres de escape a los caracteres de una cadena usando reglas de cadena de JavaScript. **Note** Esta función convertirá todas las comillas simples (`'`) en caracteres de escape (`\'`). Sin embargo, JSON no admite comillas simples con caracteres de escape. Por lo tanto, cuando la salida de esta función se utiliza en una propiedad de JSON, debe convertir todas las comillas simples con caracteres de escape (`\'`) en comillas simples normales (`'`). Esto se muestra en el siguiente ejemplo: `$util.escapeJavaScript( data).replaceAll("\\'",'''')`

Parámetro	Descripción

`$util.parseJson()`	Toma un elemento JSON en forma de cadena y devuelve una representación del resultado en forma de objeto. Puede utilizar el resultado de esta función para tener acceso y manipular los elementos de la carga de forma nativa en Apache Velocity Template Language (VTL). Por ejemplo, si tiene la siguiente carga:
	`{"errorMessage":"{\"key1\":\"var1\", \"key2\":{\"arr\":[1,2,3]}}"}`
	y usa la siguiente plantilla de asignación
	`#set ($errorMessageObj = $util.parseJson($input.path('$.errorMessage')))`
	`{`
	`"errorMessageObjKey2ArrVal" : $errorMessageObj.key2.arr[0]`
	`}`
	Obtendrá el siguiente resultado:
	`{`
	`"errorMessageObjKey2ArrVal" : 1`
	`}`

`$util.urlEncode()`	Convierte una cadena en formato "application/x-www-form-urlencoded".
`$util.urlDecode()`	Descodifica una cadena "application/x-www-form-urlencoded".
`$util.base64Encode()`	Codifica los datos en una cadena codificada en base64.

Parámetro	Descripción
`$util.base64Decode()`	Descodifica los datos de una cadena codificada en base64.

Tipos de medios binarios para las API de WebSocket en API Gateway

Actualmente, las API de WebSocket de API Gateway no admiten tramas binarias en las cargas de los mensajes entrantes. Si una aplicación cliente envía una trama binaria, API Gateway la rechaza y desconecta el cliente con el código 1003.

Existe una solución alternativa para este comportamiento. Si el cliente envía datos binarios de texto codificado (por ejemplo, base64) como una trama de texto, es posible establecer la propiedad `contentHandlingStrategy` de la integración en `CONVERT_TO_BINARY` para convertir la carga de una cadena codificada en base64 al formato binario.

Para devolver una respuesta de ruta correspondiente a una carga binaria en las integraciones que no sean de proxy, puede establecer la propiedad `contentHandlingStrategy` de la respuesta de integración en `CONVERT_TO_TEXT` para convertir la carga del formato binario a una cadena codificada en base64.

Invocación de las API de WebSocket

Después de haber implementado la API de WebSocket, las aplicaciones cliente pueden conectarse a ella y mensajes destinados a ella, y el servicio de backend puede enviar mensajes a las aplicaciones cliente conectadas:

- Puede utilizar `wscat` para conectarse a la API de WebSocket y enviar mensajes a ella para simular el comportamiento del cliente. Consulte the section called “Utilice `wscat` para conectarse y enviar mensajes a una API de WebSocket”.
- Puede utilizar la API @connections desde el servicio de backend para enviar un mensaje de devolución de llamada a un cliente conectado, obtener información sobre la conexión o desconectar el cliente. Consulte the section called “Uso de comandos de `@connections` en el servicio de backend”.
- Una aplicación cliente puede utilizar su propia biblioteca de WebSocket para invocar a la API de WebSocket.

Utilice `wscat` para conectarse y enviar mensajes a una API de WebSocket

La utilidad `wscat` es una herramienta muy práctica para probar una API de WebSocket que se ha creado e implementado en API Gateway. Puede instalar y utilizar `wscat` como se indica a continuación:

1. Descargue `wscat` de https://www.npmjs.com/package/wscat.
2. Instale `wscat` ejecutando el siguiente comando:

   ```
   npm install -g wscat
   ```

3. Para conectarse a su API, ejecute el comando `wscat` como se muestra en el siguiente ejemplo. Tenga en cuenta que en este ejemplo se presupone que `Authorization` es `NONE`.

   ```
   wscat -c wss://aabbccddee.execute-api.us-east-1.amazonaws.com/test/
   ```

   Tiene que sustituir *`aabbccddee`* por el ID real de la API, que se muestra en la consola de API Gateway o se obtiene mediante el comando `create-api` de la AWS CLI.

   Además, si la API está en una región distinta de `us-east-1`, tendrá que sustituirla por la región correcta.

4. Para probar la API, introduzca un mensaje como el siguiente mientras está conectado:

   ```
   {"{jsonpath-expression}":"{route-key}"}
   ```

   donde *`{jsonpath-expression}`* es un expresión JSONPath y *`{route-key}`* es una clave de ruta para la API. Por ejemplo:

   ```
   {"action":"action1"}
   {"message":"test response body"}
   ```

   Para obtener más información acerca de JSONPath, consulte JSONPath o JSONPath for Java.

5. Para desconectarse de la API, introduzca `ctrl-C`.

Uso de comandos de **`@connections`** en el servicio de backend

El servicio de backend puede utilizar las siguientes solicitudes HTTP de conexión de WebSocket para enviar un mensaje de devolución de llamada a un cliente conectado, obtener información sobre la conexión o desconectar el cliente.

> **Important**
>
> Estas solicitudes utilizan la autorización de IAM, por lo que debe firmarlas con Signature Version 4 (SigV4). Para hacerlo, puede usar la API de administración de API Gateway. Para obtener más información, consulte ApiGatewayManagementApi.

En el siguiente comando, tiene que sustituir *{api-id}* por el ID real de la API, que se muestra en la consola de API Gateway o se obtiene mediante el comando create-api de la AWS CLI. Debe establecer la conexión antes de utilizar este comando.

Para enviar un mensaje de devolución de llamada al cliente, utilice:

```
POST https://{api-id}.execute-api.us-east-1.amazonaws.com/{stage}/
@connections/{connection_id}
```

Puede probar esta solicitud mediante `Postman` o llamando a `awscurl`, tal y como se muestra en el siguiente ejemplo:

```
awscurl --service execute-api -X POST -d "hello world" https://{prefix}.execute-api.us-
east-1.amazonaws.com/{stage}/@connections/{connection_id}
```

Tiene que codificar el comando como URL, tal y como se muestra en el siguiente ejemplo:

```
awscurl --service execute-api -X POST -d "hello world" https://aabbccddee.execute-
api.us-east-1.amazonaws.com/prod/%40connections/R0oXAdfD0kwCH6w%3D
```

Para obtener el estado de conexión más reciente del cliente, utilice:

```
GET https://{api-id}.execute-api.us-east-1.amazonaws.com/{stage}/
@connections/{connection_id}
```

Para desconectar el cliente, utilice:

```
DELETE https://{api-id}.execute-api.us-east-1.amazonaws.com/{stage}/
@connections/{connection_id}
```

Puede crear una URL de devolución de llamada dinámicamente mediante el uso de las variables `$context` de su integración. Por ejemplo, si utiliza la integración de proxy de Lambda con una función de Lambda `Node.js`, puede crear la URL y enviar un mensaje a un cliente conectado como se indica a continuación:

```
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,
  };
};
```

Si utiliza un nombre de dominio personalizado para la API de WebSocket, quite la variable `stage` del código de la función.

Al enviar un mensaje de devolución de llamada, la función de Lambda debe tener permiso para llamar a la API de administración de puertas de enlace de API. Es posible que reciba un mensaje de error que contiene `GoneException` si publica un mensaje antes de que se establezca la conexión o después de que el cliente se haya desconectado.

Publicación de las API de WebSocket para que los clientes las invoquen

El hecho de crear y desarrollar simplemente una API de API Gateway no hace que automáticamente los usuarios puedan invocarla. Para que sea invocable, debe implementar su API en una etapa. Además, es posible que desee personalizar la URL que utilizarán los usuarios para acceder a su API. Puedes darle un dominio que sea coherente con su marca o que sea más memorable que la URL predeterminada de su API.

En esta sección, puede aprender a implementar su API y personalizar la URL que proporciona a los usuarios para acceder a ella.

> **Note**
>
> Para aumentar la seguridad de las API de API Gateway, el dominio `execute-api.{region}.amazonaws.com` se registra en la lista de sufijos públicos (PSL). Para mayor seguridad, recomendamos que utilice cookies con un prefijo `__Host-` en caso de que necesite configurar cookies confidenciales en el nombre de dominio predeterminado de las API de API Gateway. Esta práctica le ayudará a proteger su dominio de los intentos de falsificación de solicitudes entre sitios (CSRF). Para obtener más información, consulte la página de configuración de cookies en la red de desarrolladores de Mozilla.

Temas

- Creación de etapas para las API de WebSocket en API Gateway
- Implementación de las API de WebSocket en API Gateway
- Política de seguridad de las API de WebSocket en API Gateway
- Nombres de dominio personalizados para las API de WebSocket en API Gateway

Creación de etapas para las API de WebSocket en API Gateway

Una etapa de API es una referencia lógica a un estado del ciclo de vida de la API (por ejemplo, `dev`, `prod`, `beta` o `v2`). Las etapas de API se identifican por su ID de API y su nombre de etapa, y se incluyen en la URL que utiliza para invocar la API. Cada etapa es una referencia con nombre a una implementación de la API y está disponible para que las aplicaciones cliente la invoquen.

Una implementación es una instantánea de la configuración de la API. Después de implementar una API en una etapa, esta está disponible para que los clientes la invoquen. Debe implementar una API para que los cambios surtan efecto.

Variables de etapa

Las variables de etapa son pares clave-valor que se pueden definir para una etapa de una API WebSocket. Actúan como variables de entorno y se pueden usar en la configuración de la API.

Por ejemplo, puede definir una variable de etapa y, a continuación, establecer su valor como extremo HTTP para una integración de proxy HTTP. Posteriormente, puede hacer referencia al punto de conexión mediante el nombre de variable de etapa asociada. Al hacer esto, puede usar la misma configuración de API con un punto de conexión diferente en cada etapa. Del mismo modo, puede utilizar variables de etapa para especificar una integración de funciones de AWS Lambda diferente para cada etapa de su API.

> **Note**
>
> Las variables de etapa no están pensadas a fin de ser utilizadas para datos confidenciales, como credenciales. Para transferir información confidencial a las integraciones, utilice un autorizador de AWS Lambda. Puede pasar datos confidenciales a integraciones en la salida del autorizador de Lambda. Para obtener más información, consulte the section called “Formato de respuesta del autorizador de Lambda”.

Ejemplos

Para utilizar una variable de etapa para personalizar el punto de conexión de integración HTTP, primero debe establecer el nombre y el valor de la variable de etapa (por ejemplo, `url`) con un valor de `example.com`. A continuación, configure una integración de proxy HTTP. En lugar de escribir la URL del punto de conexión, puede indicar a API Gateway que use el valor de la variable de etapa,

`http://${stageVariables.url}`. Este valor indica a API Gateway que sustituya la variable de etapa `${}` en el tiempo de ejecución en función de la etapa de la API.

Puede hacer referencia a variables de etapa de una manera similar para especificar el nombre de una función de Lambda o el ARN de un rol de AWS.

A la hora de especificar el nombre de una función de Lambda como un valor de variable de etapa, debe configurar manualmente los permisos en la función de Lambda. El siguiente comando add-permission permite agregar los permisos necesarios:

```
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
```

Referencia de variables de etapa de API Gateway

URI de integración HTTP

Puede utilizar una variable de etapa como parte de una URI de integración HTTP, tal y como se muestra en los siguientes ejemplos.

- Una URI completa sin protocol – `http://${stageVariables.<variable_name>}`
- Un dominio complet – `http://${stageVariables.<variable_name>}/resource/operation`
- Un subdomini – `http://${stageVariables.<variable_name>}.example.com/resource/operation`
- Una rut – `http://example.com/${stageVariables.<variable_name>}/bar`
- Una cadena de consult – `http://example.com/foo?q=${stageVariables.<variable_name>}`

Funciones de Lambda

Puede utilizar una variable de etapa en lugar de un nombre de función de Lambda o alias, como se muestra en los ejemplos siguientes.

- `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**
>
> Para utilizar una variable de etapa para una función de Lambda, la función debe estar en la misma cuenta que la API. Las variables de etapa no admiten funciones de Lambda entre cuentas.

AWSCredenciales de integración de

Puede utilizar una variable de etapa como parte de un ARN de credenciales de usuario o rol de AWS, como se muestra en el siguiente ejemplo.

- `arn:aws:iam::<account_id>:${stageVariables.<variable_name>}`

Implementación de las API de WebSocket en API Gateway

Después de crear la API de WebSocket, debe implementarla para que los usuarios puedan invocarla.

Para implementar una API, debe crear una implementación de API y asociarla con una etapa. Cada etapa es una instantánea de la API que está disponible para que las aplicaciones cliente la invoquen.

> **Important**
>
> Cada vez que actualice una API, debe volver a implementarla. Los cambios que no sean en la configuración del escenario requieren una nueva implementación, incluidas las modificaciones en los siguientes recursos:
>
> - Rutas
> - Integraciones
> - Autorizadores
>
> De forma predeterminada, se puede incluir un máximo de 10 etapas por cada API. Se recomienda reutilizar las etapas para las implementaciones.

Para llamar a una API de WebSocket implementada, el cliente envía un mensaje a la URL de la API. La URL está determinada por el nombre de host y el nombre de etapa de la API.

> **Note**
>
> API Gateway admitirá cargas de hasta 128 KB con un tamaño máximo de trama de 32 KB. Si un mensaje supera los 32 KB, se debe dividir en varias tramas, cada una con un tamaño máximo de 32 KB.

Por ejemplo, si se utiliza el nombre de dominio predeterminado de la API, la URL de una API de WebSocket en una determinada etapa (*`{stageName}`*) tendrá el siguiente formato:

```
wss://{api-id}.execute-api.{region}.amazonaws.com/{stageName}
```

Para simplificar la URL de la API de WebSocket, puede crear un nombre de dominio personalizado (por ejemplo, `api.example.com`) para sustituir el nombre de host predeterminado de la API. El proceso de configuración es el mismo que para las API de REST. Para obtener más información, consulte the section called “Nombres de dominio personalizados”.

Las etapas permiten realizar un exhaustivo control de versiones de la API. Por ejemplo, puede implementar una API en una etapa `test` y una etapa `prod`, y utilizar la etapa `test` como compilación de pruebas y la etapa `prod` como compilación estable. Una vez que las actualizaciones superan la prueba, puede promover la etapa `test` como etapa `prod`. La promoción puede hacerse implementando de nuevo la API en la etapa `prod`. Para obtener más información sobre las etapas, consulte the section called “Configuración de una etapa”.

Temas

- Creación de una implementación de la API de WebSocket mediante la AWS CLI
- Creación de una implementación de la API de WebSocket mediante la consola de API Gateway

Creación de una implementación de la API de WebSocket mediante la AWS CLI

El siguiente comando create-deployment permite crear una implementación:

```
aws apigatewayv2 --region us-east-1 create-deployment --api-id aabbccddee
```

El resultado será similar al siguiente:

```
{
    "DeploymentId": "fedcba",
    "DeploymentStatus": "DEPLOYED",
    "CreatedDate": "2018-11-15T06:49:09Z"
}
```

La API implementada no se puede invocar hasta que la implementación se asocia a una etapa. Puede crear una etapa nueva o reutilizar una creada previamente.

El siguiente comando create-stage permite crear una nueva etapa y asociarla a la implementación:

```
aws apigatewayv2 --region us-east-1 create-stage --api-id aabbccddee --deployment-id
 fedcba --stage-name test
```

El resultado es similar al siguiente:

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

También puede reutilizar una etapa ya existente actualizando la propiedad `deploymentId` de la etapa con el ID de la implementación que acaba de crear (*`deployment-id`*). El siguiente comando update-stage permite actualizar el ID de implementación de la etapa:

```
aws apigatewayv2 update-stage --region region \
    --api-id api-id \
    --stage-name stage-name \
    --deployment-id deployment-id
```

Creación de una implementación de la API de WebSocket mediante la consola de API Gateway

Para utilizar la consola de API Gateway para crear una implementación de una API de WebSocket:

1. Inicie sesión en la consola de API Gateway y elija la API.
2. Elija Implementar API.
3. Elija la etapa que desee en el menú desplegable o escriba el nombre de una etapa nueva.

Política de seguridad de las API de WebSocket en API Gateway

API Gateway aplica una política de seguridad de TLS_1_2 para todos los puntos de conexión de la API de WebSocket.

Una política de seguridad es una combinación predefinida de versión mínima de TLS y conjuntos de cifrado que ofrece Amazon API Gateway. El protocolo TLS soluciona los problemas de seguridad de red como, por ejemplo, la manipulación y el acceso no autorizado entre un cliente y el servidor. Cuando sus clientes establecen un protocolo de conexión a TLS con la API a través del dominio personalizado, la política de seguridad aplica la versión de TLS y opciones del conjunto de cifrado que sus clientes pueden elegir utilizar. Esta política de seguridad acepta el tráfico de TLS 1.2 y TLS 1.3 y rechaza el tráfico de TLS 1.0.

Protocolos y cifrados TLS compatibles para las API de WebSocket

En la siguiente tabla se describen los protocolos TLS compatibles con las API de WebSocket.

Protocolos TLS	Política de seguridad de TLS_1_2
TLSv1.3	Sí
TLSv1.2	Sí

En la siguiente tabla se describen los cifrados de TLS que están disponibles para la política de seguridad de TLS 1_2 para las API de WebSocket.

Cifrados TLS	Política de seguridad de TLS_1_2
TLS_AES_128_GCM_SHA256	Sí
TLS_AES_256_GCM_SHA384	Sí
TLS_CHACHA20_POLY1305_SHA256	Sí
ECDHE-ECDSA-AES128-GCM-SHA256	Sí
ECDHE-RSA-AES128-GCM-SHA256	Sí
ECDHE-ECDSA-AES128-SHA256	Sí
ECDHE-RSA-AES128-SHA256	Sí
ECDHE-ECDSA-AES256-GCM-SHA384	Sí
ECDHE-RSA-AES256-GCM-SHA384	Sí
ECDHE-ECDSA-AES256-SHA384	Sí
ECDHE-RSA-AES256-SHA384	Sí
AES128-GCM-SHA256	Sí
AES128-SHA256	Sí
AES256-GCM-SHA384	Sí
AES256-SHA256	Sí

Nombre del cifrado OpenSSL y RFC

OpenSSL e IETF RFC 5246 utilizan nombres distintos para los mismos cifrados. Para obtener una lista de los nombres de los cifrados, consulte the section called “Nombre del cifrado OpenSSL y RFC”.

Información acerca de las API de REST y las API HTTP

Para obtener más información sobre las API de REST y las API HTTP, consulte the section called "Elección de una política de seguridad" y the section called "Política de seguridad de las API de HTTP en API Gateway".

Nombres de dominio personalizados para las API de WebSocket en API Gateway

Losnombres de dominio personalizados son direcciones URL más sencillas e intuitivas que puede proporcionar a los usuarios de la API.

Después de implementar la API, usted (y sus clientes) puede invocar la API utilizando la URL base predeterminada con el siguiente formato:

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

donde *api-id* lo genera API Gateway, *region* es la región de AWS y *stage* lo especifica usted al implementar la API.

La parte del nombre de host de la URL, *api-id*`.execute-api.`*region*`.amazonaws.com`, hace referencia a un punto de conexión de la API. El punto de conexión de la API predeterminado se genera aleatoriamente y no es muy descriptivo.

Con los nombres de dominio personalizados, puede configurar el nombre de host de la API y elegir una ruta base (por ejemplo, `myservice`) para asignarle una URL alternativa. Por ejemplo, una URL base de la API más simple puede ser:

```
https://api.example.com/myservice
```

Consideraciones

Las siguientes consideraciones pueden afectar el uso de un nombre de dominio personalizado.

- Si asigna un nombre de dominio personalizado a una API de WebSocket, no puede asignarlo a una API de REST o HTTP.
- Solo se admiten nombres de dominio personalizados regionales.
- Para la versión de TLS mínima, solo se admite TLS 1.2.

- Debe crear o actualizar el registro de recursos del proveedor de DNS para asignarlo al punto de conexión de la API. Si no se realiza este mapeo, las solicitudes de API vinculadas al nombre de dominio personalizado no pueden llegar a API Gateway.
- Puede admitir un número casi infinito de nombres de dominio sin exceder el límite predeterminado mediante el uso de un certificado comodín. Para obtener más información, consulte the section called “Nombres de dominio personalizados comodín”.

Requisitos previos

Los siguientes son requisitos previos para un nombre de dominio personalizado.

Registrar un nombre de dominio

Debe disponer de un nombre de dominio de Internet registrado para poder crear nombres de dominio personalizados para sus API. Puede registrar su nombre de dominio de Internet con Amazon Route 53 o un registrador de dominios de terceros de su elección. Su nombre de dominio personalizado puede ser el nombre de un subdominio o el dominio raíz (lo que recibe el nombre de “vértice de zona”) de un dominio de Internet registrado.

El nombre de dominio debe seguir la especificación RFC 1035 y puede tener un máximo de 63 octetos por etiqueta y 255 octetos en total.

Certificados para nombres de dominio personalizados

Antes de configurar un nombre de dominio personalizado para una API, debe disponer de un certificado SSL/TLS listo en ACM. Si ACM no está disponible en la región de AWS donde va a crear su nombre de dominio personalizado, debe importar un certificado en API Gateway en esa región.

Para importar un certificado SSL/TLS, debe proporcionar el cuerpo del certificado SSL/TLS con formato PEM, su clave privada y la cadena de certificados para el nombre de dominio personalizado.

Los certificados almacenados en ACM se identifican mediante su ARN. Con los certificados emitidos por ACM no tiene que preocuparse de si se exponen datos del certificado confidenciales, como la clave privada. Para utilizar un certificado administrado por AWS para un nombre de dominio, solo tiene que hacer referencia a su ARN.

Si la aplicación utiliza asignación de certificados, también conocido como asignación SSL, para asignar un certificado de ACM, es posible que la aplicación no se pueda conectar al dominio una vez que AWS renueva el certificado. Para obtener más información, consulte Problemas de asignación de certificados en la Guía del usuario de AWS Gestor de certificados.

Nombres de dominio personalizados comodín

Con los nombres de dominio personalizados comodín, puede admitir un número casi infinito de nombres de dominio sin exceder el límite predeterminado. Por ejemplo, puede dar a cada uno de los clientes su propio nombre de dominio, *customername*`.api.example.com`.

Para crear un nombre de dominio personalizado comodín, especifique un comodín (`*`) como primer subdominio de un dominio personalizado que represente todos los subdominios posibles de un dominio raíz.

Por ejemplo, el nombre de dominio personalizado comodín `*.example.com` produce subdominios como `a.example.com`, `b.example.com` y `c.example.com`, que enrutan al mismo dominio.

Los nombres de dominio personalizados comodín admiten configuraciones distintas de los nombres de dominio personalizados estándar de API Gateway. Por ejemplo, en una sola cuenta de AWS, puede configurar `*.example.com` y `a.example.com` para que se comporten de manera diferente.

Puede utilizar las variables de contexto `$context.domainName` y `$context.domainPrefix` para determinar el nombre de dominio que un cliente utilizó para llamar a la API. Para obtener más información acerca de las variables de contexto, consulte Variables para transformaciones de datos para API Gateway.

Para crear un nombre de dominio personalizado comodín, debe proporcionar un certificado emitido por ACM que se haya validado utilizando el método DNS o de validación de correo electrónico.

> **Note**
>
> No se puede crear un nombre de dominio personalizado comodín si una cuenta de AWS diferente ha creado un nombre de dominio personalizado que entra en conflicto con el nombre de dominio personalizado comodín. Por ejemplo, si la cuenta A ha creado `a.example.com`, la cuenta B no puede crear el nombre de dominio personalizado comodín `*.example.com`.
> Si la cuenta A y la cuenta B comparten un propietario, puede contactar con el Centro de soporte de AWS para solicitar una excepción.

Siguientes pasos para nombres de dominio personalizados

Para configurar un nombre de dominio personalizado para una API de HTTP, utilice la sección API de REST de la Guía para desarrolladores de API Gateway.

En primer lugar, especifique el certificado para el nombre de dominio personalizado. Para obtener más información, consulte the section called “Preparación de certificados en AWS Gestor de certificados”. A continuación, cree un nombre de dominio personalizado regional. Para obtener más información, consulte the section called “Configuración de un nombre de dominio personalizado regional”.

Asignación de etapas de API a un nombre de dominio personalizado para las API de WebSocket

Los mapeos de la API se utilizan para conectar etapas de la API a un nombre de dominio personalizado. Después de crear un nombre de dominio y configurar registros DNS, se utilizan mapeos de la API para enviar tráfico a las API a través del nombre de dominio personalizado.

Un mapeo de la API especifica una API, una etapa y, de forma opcional, una ruta que se utilizará para el mapeo. Por ejemplo, puede mapear la etapa de `production` de una API a `wss://api.example.com/orders`.

Antes de crear un mapeo de la API, debe tener una API, una etapa y un nombre de dominio personalizado. Para obtener más información sobre cómo crear un nombre de dominio personalizado, consulte the section called “Configuración de un nombre de dominio personalizado regional”.

Restricciones

- En un mapeo de la API, el nombre de dominio personalizado y las API mapeadas deben estar en la misma cuenta de AWS.
- Los mapeos de la API deben contener solo letras, números y los siguientes caracteres: `$-_.+!*'()`.
- La longitud máxima de la ruta en un mapeo de la API es de 300 caracteres.
- No puede mapear las API de WebSocket al mismo nombre de dominio personalizado que una API HTTP o API REST.
- Si crea una asignación de API con varios niveles, API Gateway convierte todos los nombres de encabezado a minúsculas.

Creación de un mapeo de la API

Para crear un mapeo de la API, primero debe crear un nombre de dominio personalizado, una API y una etapa. Para obtener información sobre cómo crear un nombre de dominio personalizado, consulte the section called “Configuración de un nombre de dominio personalizado regional”.

Consola de administración de AWS

Para crear un mapeo de la API

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Elija Custom domain names (Nombres de dominio personalizados).
3. Seleccione un nombre de dominio personalizado que ya haya creado.
4. Elija API mappings (Mapeos de la API).
5. Elija Configure API mappings (Configurar asignaciones de API).
6. Seleccione Add new mapping (Agregar nueva asignación).
7. Introduzca una API, una Stage (Etapa) y, de forma opcional, una Path (Ruta).
8. Seleccione Guardar.

AWS CLI

El siguiente comando create-api-mapping permite crear una asignación de API. En este ejemplo, API Gateway envía solicitudes a `api.example.com/v1` a la API y a la etapa especificada.

```
aws apigatewayv2 create-api-mapping \
    --domain-name api.example.com \
    --api-mapping-key v1 \
    --api-id a1b2c3d4 \
    --stage test
```

CloudFormation

El siguiente ejemplo de CloudFormation crea un mapeo de la API.

```
MyApiMapping:
  Type: 'AWS::ApiGatewayV2::ApiMapping'
  Properties:
```

```
    DomainName: api.example.com
    ApiMappingKey: 'v1'
    ApiId: !Ref MyApi
    Stage: !Ref MyStage
```

Tipos de direcciones IP para nombres de dominio personalizados para API de WebSocket

Al crear un nombre de dominio personalizado, se especifica el tipo de direcciones IP que puede invocar el dominio. Puede elegir IPv4 para resolver las direcciones IPv4 para invocar el dominio o puede elegir pila doble para permitir que las direcciones IPv4 e IPv6 invoquen el dominio. Le recomendamos que establezca el tipo de dirección IP en pila doble para mitigar el agotamiento del espacio IP o para mejorar la postura de seguridad. Para obtener más información sobre los beneficios de un tipo de dirección IP de pila doble, consulte IPv6 en AWS.

Consideraciones para los tipos de direcciones IP

Es posible que las siguientes consideraciones afecten al uso de los tipos de direcciones IP.

- El tipo de dirección IP predeterminado para los nombres de dominio personalizados de API Gateway es IPv4.
- El nombre de dominio personalizado no necesita tener el mismo tipo de dirección IP para todas las API asignadas a él. Si desactiva el punto de conexión de la API predeterminado, esto podría afectar a la forma en que los intermediarios pueden invocar a la API.

Cambio del tipo de dirección IP del nombre de dominio personalizado

Puede cambiar el tipo de dirección IP actualizando la configuración de punto de conexión del nombre de dominio. Puede actualizar la configuración de punto de conexión mediante el uso de la Consola de administración de AWS, la AWS CLI, CloudFormation, o un AWS SDK.

Consola de administración de AWS

Cambio del tipo de dirección IP de un nombre de dominio personalizado

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Elija un nombre de dominio personalizado público.

3. Elija la Configuración de punto de conexión.
4. Para tipo de dirección IP, seleccione IPv4 o Pila doble.
5. Seleccione Save.

AWS CLI

El siguiente comando update-domain-name actualiza una API para que tenga un tipo de dirección IP de pila doble:

```
aws apigatewayv2 update-domain-name \
   --domain-name dualstack.example.com \
   --domain-name-configurations CertificateArn=arn:aws:acm:us-
east-1:111122223333:certificate/abcd1234-5678-abc,IpAddressType=dualstack
```

El resultado será similar al siguiente:

```
{
    "ApiMappingSelectionExpression": "$request.basepath",
    "DomainName": "dualstack.example.com",
    "DomainNameConfigurations": [
        {
            "ApiGatewayDomainName": "d-abcd1234.execute-api.us-
east-1.amazonaws.com",
            "CertificateArn": "arn:aws:acm:us-east-1:111122223333:certificate/
abcd1234-5678-abc",
            "DomainNameStatus": "AVAILABLE",
            "EndpointType": "REGIONAL",
            "HostedZoneId": "Z3LQWSYCGH4ADY",
            "SecurityPolicy": "TLS_1_2",
            "IpAddressType": "dualstack"
        }
    ],
    "Tags": {}
}
```

Deshabilitación del punto de conexión predeterminado para las API de WebSocket

De forma predeterminada, los clientes pueden invocar su API mediante el punto de conexión `execute-api` que API Gateway genera para ella. Para asegurarse de que los clientes solo puedan

acceder a su API mediante un nombre de dominio personalizado, deshabilite el punto de conexión predeterminado `execute-api`. Cuando deshabilita el punto de conexión predeterminado, esto afecta a todas las etapas de una API.

El siguiente procedimiento muestra cómo deshabilitar el punto de conexión predeterminado para una API de WebSocket.

Consola de administración de AWS

1. Inicie sesión en la consola de API Gateway, en https://console.aws.amazon.com/apigateway.
2. Elija una API de WebSocket.
3. Elija Configuración de la API.
4. En Detalles de API, elija Editar.
5. En Punto de conexión predeterminado, elija Inactivo.
6. Seleccione Save changes (Guardar cambios).
7. En el panel de navegación principal, elija Rutas.
8. Elija Implementar y, a continuación, vuelva a implementar la API o cree una nueva etapa para que el cambio surta efecto.

AWS CLI

El siguiente comando update-api desactiva el punto de conexión predeterminado de una API de WebSocket.

```
aws apigatewayv2 update-api \
    --api-id abcdef123 \
    --disable-execute-api-endpoint
```

Después de desactivar el punto de conexión predeterminado, debe implementar la API para que el cambio surta efecto.

El siguiente comando de la AWS CLI crea una implementación.

```
aws apigatewayv2 create-deployment \
    --api-id abcdef123 \
    --stage-name dev
```

Protección de las API de WebSocket en API Gateway

API Gateway proporciona una serie de formas de proteger su API de ciertas amenazas, como usuarios malintencionados o picos de tráfico. Puede proteger la API mediante estrategias como generar certificados SSL o establecer objetivos de limitación. Para obtener más información acerca de cómo generar certificados SSL, consulte the section called “Certificados generados por API Gateway”. El resto de esta sección trata sobre cómo establecer objetivos de limitación.

Puede configurar la limitación de las API para evitar que se vean desbordadas por un número excesivo de solicitudes. La limitación se aplica como medida compensatoria y se debe considerar como un valor objetivo, más que como un límite de solicitudes garantizado.

API Gateway limita las solicitudes a la API utilizando el algoritmo de bucket de tokens, donde un token cuenta por una solicitud. En concreto, API Gateway examina el ratio de solicitudes y una ráfaga de envíos de solicitudes para todas las API de su cuenta, por región. En el algoritmo de bucket de tokens, una ráfaga puede permitir que se sobrepasen los límites predefinidos, pero también hay otros factores que pueden hacer que se sobrepasen los límites en algunos casos.

Cuando los envíos de solicitudes superan los límites de ratio de solicitudes en estado estable y de ráfaga, API Gateway comienza a limitar las solicitudes. Los clientes pueden recibir respuestas de error `429 Too Many Requests` en este momento. Tras capturar estas excepciones, el cliente puede reenviar las solicitudes que han producido un error de forma que limite el ratio.

Como desarrollador de la API, puede configurar los límites objetivos para las etapas o rutas individuales de la API con el fin de mejorar el rendimiento general de todas las API de su cuenta.

Limitación de nivel de cuenta por región

De forma predeterminada, API Gateway limita las solicitudes de estado constante por segundo (RPS) en todas las API de una cuenta de AWS, por región. También limita la ráfaga (es decir, el tamaño máximo del bucket) en todas las API dentro de una cuenta de AWS, por región. En API Gateway, el límite de ráfaga representa el número máximo de envíos de solicitudes simultáneas que API Gateway; abordará antes de devolver respuestas de error `429 Too Many Requests`. Para obtener más información sobre las cuotas de limitación, consulte Cuotas y notas importantes.

Los límites por cuenta se aplican a todas las API de una cuenta en una región determinada. El límite de ratio de nivel de cuenta se puede aumentar previa solicitud. Los límites más altos son posibles con API que tienen tiempos de espera más cortos y cargas útiles más pequeñas. Para solicitar un aumento de los límites de limitación de nivel de cuenta por región, contacte con el Centro de soporte

de AWS. Para obtener más información, consulte Cuotas y notas importantes. Tenga en cuenta que estos límites no pueden ser superiores a los límites de limitación de AWS.

Limitación de nivel de ruta

Puede definir la limitación de nivel de ruta para invalidar los límites de limitación de las solicitudes de nivel de cuenta de una determinada etapa o de las rutas individuales de la API. Los límites de limitación de rutas predeterminados no pueden superar los límites de ratio a nivel de cuenta.

Puede configurar la limitación de nivel de ruta mediante la AWS CLI. El siguiente comando update-stage permite configurar la limitación de nivel de ruta para la etapa y la ruta especificadas de una API.

```
aws apigatewayv2 update-stage \
    --api-id a1b2c3d4 \
    --stage-name dev \
    --route-settings '{"messages":
{"ThrottlingBurstLimit":100,"ThrottlingRateLimit":2000}}'
```

Supervisión de las API de WebSocket en API Gateway

Puede utilizar las métricas de CloudWatch y CloudWatch Logs para monitorear las API de WebSocket. Al combinar registros y métricas, puede registrar errores y monitorear el rendimiento de su API.

> **Note**
>
> API Gateway podría no generar registros y métricas en los siguientes casos:
>
> - Errores 413: Entidad de solicitud demasiado grande
> - Errores 429: Demasiadas Solicitudes
> - Errores de la serie 400 de solicitudes enviadas a un dominio personalizado que no tiene asignación de API
> - Errores de la serie 500 causados por errores internos

Temas

- Supervisión de la ejecución de la API de WebSocket con métricas de CloudWatch

- Configuración del registro de las API de WebSocket en API Gateway

Supervisión de la ejecución de la API de WebSocket con métricas de CloudWatch

Puede usar métricas de Amazon CloudWatch para monitorear las API de WebSocket. La configuración es similar a la utilizada para las API de REST. Para obtener más información, consulte Supervisión de la ejecución de la API de REST con métricas de Amazon CloudWatch.

Las siguientes métricas se admiten para las API de WebSocket:

Métrica	Descripción
ConnectCount	El número de mensajes enviados a la integración de ruta `$connect`.
MessageCount	El número de mensajes enviados a la API de WebSocket, con origen o destino en el cliente.
IntegrationError	El número de solicitudes que devuelven una respuesta 4XX/5XX desde la integración.
ClientError	El número de solicitudes para las que API Gateway devuelve una respuesta 4XX antes de que se invoque la integración.
ExecutionError	Errores que se han producido al llamar a la integración.
IntegrationLatency	El tiempo que transcurre entre el momento en que API Gateway envía la solicitud a la integración y el momento

Métrica	Descripción
	en que API Gateway recibe la respuesta de la integración. Se suprime para devoluciones de llamada e integraciones simuladas.

Puede usar las dimensiones de la tabla siguiente para filtrar las métricas de API Gateway.

Dimensión	Descripción
ApiId	Filtra las métricas de API Gateway para una API con el ID de API especificado.
ApiId, Stage	Filtra las métricas de API Gateway para una etapa de API cuyos ID de API y de etapa se hayan especificado.
ApiId, método, recurso, etapa	Filtra las métricas de API Gateway para un método de API con el ID de API, el ID de etapa, la ruta de recursos y el ID de enrutamiento específicos. API Gateway no enviará estas métricas a menos que haya habilitado explícitamente las métricas detalladas de CloudWatch. Puede hacerlo mediante una llamada a la acción UpdateStage de la API de REST de API Gateway V2 para actualizar la propiedad

Dimensión	Descripción
	`detailedMetricsEnabled` en `true`. También puede llamar al comando update-stage AWS CLI para actualizar la propiedad `DetailedMetricsEnabled` a `true`. Si activa estas métricas, se le cobrarán cargos adicionales en su cuenta. Para obtener más información sobre precios, consulte Precios de Amazon CloudWatch.

Configuración del registro de las API de WebSocket en API Gateway

Puede habilitar el registro para escribir registros en CloudWatch Logs. Existen dos tipos de registro de API en CloudWatch: los registros de ejecución y los registros de acceso. En el registro de ejecución, API Gateway administra los CloudWatch Logs. El proceso incluye la creación de grupos y flujos de registros y la notificación a los flujos de registro de las solicitudes y respuestas del intermediario.

Para aumentar la seguridad, le recomendamos que utilice el registro de ejecución en el nivel `INFO` o `ERROR`. Es probable que deba hacerlo para asegurar el cumplimiento de diversos marcos normativos. Para obtener más información, consulte Controles de Amazon API Gateway en la Guía del usuario de AWS Security Hub.

En los registros de acceso, tanto usted como el desarrollador de la API pretenden registrar quién ha obtenido acceso a la API y cómo el intermediario ha obtenido acceso a la API. Puede crear su propio grupo de registros o elegir un grupo de registros existente, que podría administrarse a través de API Gateway. Para especificar los detalles de acceso, seleccione variables `$context` (expresadas en un formato que elija) y elija un grupo de registro como destino.

Para ver instrucciones sobre cómo configurar el registro de CloudWatch, consulte the section called “Configuración del registro de acceso mediante la consola de API Gateway”.

Al especificar el Log Format (Formato de registro), puede elegir las variables de contexto que deben registrarse. Se admiten las siguientes variables.

Parámetro	Descripción
`$context.apiId`	El identificador que API Gateway asigna a su API.
`$context.authorize.error`	El mensaje de error de autorización.
`$context.authorize.latency`	La latencia de autorización en ms.
`$context.authorize.status`	El código de estado devuelto por un intento de autorización.
`$context.authorizer.error`	El mensaje de error devuelto por un autorizador.
`$context.authorizer.integrationLatency`	La latencia del autorizador de Lambda en ms.
`$context.authorizer.integrationStatus`	El código de estado que devuelve un autorizador de Lambda.
`$context.authorizer.latency`	La latencia del autorizador en ms.
`$context.authorizer.requestId`	El ID de solicitud del punto de conexión de AWS.
`$context.authorizer.status`	El código de estado devuelto por un autorizador.
`$context.authorizer.principalId`	La identificación de usuario principal que está asociada con el token que envía el cliente y devuelve la función de Lambda del autorizador personalizado de Lambda de API Gateway. (El autorizador de Lambda se conocía anteriormente como “autorizador personalizado”).

Parámetro	Descripción
`$context.authorizer.` *`property`*	El valor en forma de cadena del par clave-valor especificado de la asignación `context` que devuelve la función de Lambda del autorizador de Lambda de API Gateway. Por ejemplo, si el autorizador devuelve la siguiente asignación de `context`: `"context" : { "key": "value", "numKey": 1, "boolKey": true }` la llamada a `$context.authorizer.key` devuelve la cadena `"value"`, la llamada a `$context.authorizer.numKey` devuelve la cadena `"1"` y la llamada a `$context.authorizer.boolKey` devuelve la cadena `"true"`.
`$context.authenticate.error`	El mensaje de error devuelto por un intento de autorización.
`$context.authenticate.latency`	La latencia de autenticación en ms.
`$context.authenticate.status`	El código de estado devuelto por un intento de autenticación.
`$context.connectedAt`	Hora de la conexión en formato de tiempo Unix.
`$context.connectionId`	Identificador único para la conexión que se puede utilizar para realizar una devolución de llamada al cliente.

Parámetro	Descripción
`$context.domainName`	Nombre de dominio para la API de WebSocket . Se puede utilizar para realizar una devolució n de llamada al cliente (en lugar de un valor de código rígido).
`$context.error.message`	Una cadena que contiene un mensaje de error de API Gateway.
`$context.error.messageString`	El valor entrecomillado de `$context. error.message` , es decir, `"$context .error.message"` .
`$context.error.responseType`	El tipo de respuesta de error.
`$context.error.validationEr rorString`	Una cadena que contiene un mensaje de error de validación detallado.
`$context.eventType`	El tipo de evento: `CONNECT`, `MESSAGE` o `DISCONNECT` .
`$context.extendedRequestId`	Es igual que `$context.requestId` .
`$context.identity.accountId`	El ID de cuenta de AWS asociado con la solicitud.
`$context.identity.apiKey`	Clave del propietario de API asociada a la solicitud de API habilitada para claves.
`$context.identity.apiKeyId`	ID de clave de API asociado a la solicitud de API habilitada para claves
`$context.identity.caller`	El identificador principal del intermediario que firmó la solicitud. Compatible con rutas que utilizan la autorización de IAM.

Parámetro	Descripción
`$context.identity.cognitoAuthenticationProvider`	Una lista separada por comas de todos los proveedores de autenticación de Amazon Cognito utilizados por el intermediario que realiza la solicitud. Solo está disponible si la solicitud se firmó con las credenciales de Amazon Cognito. Por ejemplo, para una identidad de un grupo de usuarios de Amazon Cognito, `cognito-idp.` *`region`*`.amazonaws.com/` *`user_pool_id`* `,cognito-idp.` *`region`*`.amazonaws.com/` *`user_pool_id`* `:CognitoSignIn:` *`token subject claim`* Consulte Uso de las identidades federadas en la Guía para desarrolladores de Amazon Cognito para obtener información sobre los proveedores de autenticación de Amazon Cognito disponibles.
`$context.identity.cognitoAuthenticationType`	El tipo de autenticación de Amazon Cognito del intermediario que realiza la solicitud. Solo está disponible si la solicitud se firmó con las credenciales de Amazon Cognito. Los valores posibles incluyen `authenticated` para identidades autenticadas y `unauthenticated` para identidades no autenticadas.
`$context.identity.cognitoIdentityId`	El ID de identidad de Amazon Cognito del intermediario que realiza la solicitud. Solo está disponible si la solicitud se firmó con las credenciales de Amazon Cognito.

Parámetro	Descripción
`$context.identity.cognitoId entityPoolId`	El ID del grupo de identidades de Amazon Cognito del intermediario que realiza la solicitud. Solo está disponible si la solicitud se firmó con las credenciales de Amazon Cognito.
`$context.identity.principalOrgId`	El ID de organización de AWS. Compatible con rutas que utilizan la autorización de IAM.
`$context.identity.sourceIp`	La dirección IP de origen de la conexión TCP que realiza la solicitud de API Gateway.
`$context.identity.user`	El identificador principal del usuario que se autorizará a acceder al recurso. Compatible con rutas que utilizan la autorización de IAM.
`$context.identity.userAgent`	El agente de usuario del intermediario de la API.
`$context.identity.userArn`	El Nombre de recurso de Amazon (ARN) del usuario identificado después de la autentica ción.
`$context.integration.error`	El mensaje de error devuelto por una integraci ón.
`$context.integration.integr ationStatus`	Para la integración de proxy de Lambda, el código de estado que se devuelve desde AWS Lambda, y no desde el código de función de Lambda del backend.
`$context.integration.latency`	La latencia de integración en ms. Es igual que `$context.integrationLatency` .
`$context.integration.requestId`	El ID de solicitud del punto de conexión de AWS. Es igual que `$context.awsEndpoi ntRequestId` .

Parámetro	Descripción
`$context.integration.status`	El código de estado devuelto por una integración. Para integraciones de proxy de Lambda, este es el código de estado que devuelve su código de la función de Lambda. Es igual que `$context.integrationStatus` .
`$context.integrationLatency`	La latencia de integración en ms, disponible únicamente para el registro de acceso.
`$context.messageId`	Un ID único del lado del servidor para un mensaje. Solo está disponible si `$context.eventType` es `MESSAGE`.
`$context.requestId`	Igual que `$context.extendedRequestId` .
`$context.requestTime`	Hora de la solicitud en formato CLF-(`dd/MMM/yyyy:HH:mm:ss +-hhmm`).
`$context.requestTimeEpoch`	Hora de la solicitud en formato Epoch en milisegundos.
`$context.routeKey`	La clave de ruta seleccionada.
`$context.stage`	La etapa de implementación de la llamada a la API (por ejemplo, beta o prod).
`$context.status`	Estado de la respuesta.
`$context.waf.error`	El mensaje de error devuelto por AWS WAF.
`$context.waf.latency`	La latencia de AWS WAF en ms.
`$context.waf.status`	El código de estado devuelto por AWS WAF.

Algunos ejemplos de los formatos de registro de acceso que se utilizan habitualmente se muestran en la consola de API Gateway y se detallan a continuación.

- `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
```

  Los caracteres de continuación (\) deben entenderse como una ayuda visual. El formato de registro debe ser una sola línea. Puede agregar un carácter de nueva línea (\n) al final del formato de registro para incluir una nueva línea al final de cada entrada de registro.

- `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"
}
```

  Los caracteres de continuación (\) deben entenderse como una ayuda visual. El formato de registro debe ser una sola línea. Puede agregar un carácter de nueva línea (\n) al final del formato de registro para incluir una nueva línea al final de cada entrada de registro.

- `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>
```

Los caracteres de continuación (\) deben entenderse como una ayuda visual. El formato de registro debe ser una sola línea. Puede agregar un carácter de nueva línea (\n) al final del formato de registro para incluir una nueva línea al final de cada entrada de registro.

- CSV (valores separados por comas):

```
$context.identity.sourceIp,$context.identity.caller, \
$context.identity.user,$context.requestTime,$context.eventType, \
$context.routeKey,$context.connectionId,$context.status, \
$context.requestId
```

Los caracteres de continuación (\) deben entenderse como una ayuda visual. El formato de registro debe ser una sola línea. Puede agregar un carácter de nueva línea (\n) al final del formato de registro para incluir una nueva línea al final de cada entrada de registro.

Referencia del nombre de recurso de Amazon (ARN) de API Gateway

En las tablas siguientes se enumeran los nombres de recurso de Amazon (ARN) correspondientes a los recursos de API Gateway. Para obtener más información sobre el uso de ARN en políticas de AWS Identity and Access Management, consulte Cómo funciona Amazon API Gateway con IAM y Control del acceso a una API de REST con permisos de IAM.

Recursos de API HTTP y API de WebSocket

Recurso	ARN
AccessLogSettings	`arn:partition :apigateway: region::/apis/api-id/stages/stage-name /accesslogsettings`
API	`arn:partition :apigateway: region::/apis/api-id`
Apis	`arn:partition :apigateway: region::/apis`
Autorizador	`arn:partition :apigateway: region::/apis/api-id/authorizers/ id`
Autorizadores	`arn:partition :apigateway: region::/apis/api-id/authorizers`
Cors	`arn:partition :apigateway: region::/apis/api-id/cors`

Recurso	ARN
Implementación	arn:*partition* :apigateway: *region*::/apis/*api-id*/deployments/ *id*
Implementaciones	arn:*partition* :apigateway: *region*::/apis/*api-id*/deployments
ExportedAPI	arn:*partition* :apigateway: *region*::/apis/*api-id*/exports/ *specification*
Integración	arn:*partition* :apigateway: *region*::/apis/*api-id*/integrations/ *integration-id*
Integraciones	arn:*partition* :apigateway: *region*::/apis/*api-id*/integrations
IntegrationResponse	arn:*partition* :apigateway: *region*::/apis/*api-id*/integrationresponses/ *integration-response*
IntegrationResponses	arn:*partition* :apigateway: *region*::/apis/*api-id*/integrationresponses
Modelo	arn:*partition* :apigateway: *region*::/apis/*api-id*/models/*id*
Modelos	arn:*partition* :apigateway: *region*::/apis/*api-id*/models

Recurso	ARN
ModelTemplate	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/models/*id*/ template
Ruta	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/routes/*id*
Rutas	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*
Etapa	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/ stages/*stage-name*
Etapas	arn:*partition* :apigatew ay: *region*::/apis/*api-id*/stages
VpcLink	arn:*partition* :apigatew ay: *region*::/vpclinks/ *vpclink-id*

Recurso	ARN
VpcLinks	arn:*partition* :apigatew ay: *region*::/vpclinks

Recursos de API de REST

Recurso	ARN
Cuenta	arn:*partition* :apigatew ay: *region*::/account
ApiKey	arn:*partition* :apigatew ay: *region*::/apikeys/ *id*
ApiKeys	arn:*partition* :apigatew ay: *region*::/apikeys
Autorizador	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ authorizers/ *id*
Autorizadores	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ authorizers
ClientCertificate	arn:*partition* :apigatew ay: *region*::/clientcertifica tes/ *id*
ClientCertificates	arn:*partition* :apigatew ay: *region*::/clientcertificates
Implementación	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ deployments/ *id*

Recurso	ARN
Implementaciones	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*
DocumentationVersions	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ documentation/versions
GatewayResponse	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ gatewayresponses/ *response-type*
GatewayResponses	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ gatewayresponses
Integración	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ resources/ *resource-id* /methods/ *http-method* /integration

Recurso	ARN
IntegrationResponse	arn:*partition* :apigateway: *region*::/restapis/ *api-id*/resources/ *resource-id* /methods/ *http-method* /integration/responses/ *status-code*
Método	arn:*partition* :apigateway: *region*::/restapis/ *api-id*/resources/ *resource-id* /methods/ *http-method*
MethodResponse	arn:*partition* :apigateway: *region*::/restapis/ *api-id*/resources/ *resource-id* /methods/ *http-method* /responses/ *status-code*
Modelo	arn:*partition* :apigateway: *region*::/restapis/ *api-id*/models/*model-name*
Modelos	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
Recurso	arn:*partition* :apigateway: *region*::/restapis/ *api-id*/resources/ *id*

Recurso	ARN
Recursos	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ resources
RestApi	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*
RestApis	arn:*partition* :apigatew ay: *region*::/restapis
Etapa	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ stages/*stage-name*
Etapas	arn:*partition* :apigatew ay: *region*::/restapis/ *api-id*/ stages
Etiquetas	arn:*partition* :apigatew ay: *region*::/tags/*url-encoded-resource-arn*
Plantilla	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*

Recurso	ARN
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

Recursos de nombres de dominio

Recurso	ARN
ApiMapping	arn:*partition* :apigatew ay: *region*::/domainnames/ *domain-na me* /apimappings/ *id*
ApiMappings	arn:*partition* :apigatew ay: *region*::/domainnames/ *domain-na me* /apimappings
BasePathMapping	arn:*partition* :apigatew ay: *region*::/domainnames/ *domain-na me* /basepathmappings/ *basepath*
BasePathMappings	arn:*partition* :apigatew ay: *region*::/domainnames/ *domain-na me* /basepathmappings
DomainName	arn:*partition* :apigatew ay: *region*::/domainnames/ *domain-na me*

Recurso	ARN
DomainNameAccessAssociation	arn:*partition* :apigateway: *region*:*account-id* :/domainnameaccessassociations/domainname/ *domain-name* /vpcesource/ *vpce-source-id*
DomainNameAccessAssociations	arn:*partition* :apigateway: *region*:*account-id* :/domainnameaccessassociations
DomainNames	arn:*partition* :apigateway: *region*::/domainnames
RoutingRule	arn:*partition* :apigateway: *region*:*account-id* :/domainnames/ *domain-name* /routingrules/ *routing-rule-id*

execute-api (API HTTP, API de WebSocket y API de REST)

Recurso	ARN
Punto de conexión de API de WebSocket	arn:*partition* :execute-api: *region*:*account-id* :*api-id*/*stage*/*route-key*
Punto de conexión de API HTTP y de API de REST	arn:*partition* :execute-api: *region*:*account-id* :*api-id*/*stage*/*http-method* /*resource-path*
Autorizador de Lambda **	arn:*partition* :execute-api: *region*:*account-id* :*api-id*/authorizers/ *authorizer-id*

* El ARN del punto de conexión de la ruta `$default` para las API de HTTP es `arn:partition:execute-api:region:account-id:api-id/*/$default`.

** Este ARN solo se aplica cuando se establece la condición `SourceArn` en la política de recurso para una función de autorizador de Lambda. Para ver un ejemplo, consulte the section called “Crear un autorizador de Lambda”.

Extensiones de OpenAPI para API Gateway

Las extensiones de API Gateway admiten la autorización específica de AWS y las integraciones de API específicas de API Gateway para las API REST y las API HTTP. En esta sección, describimos las extensiones de API Gateway para la especificación de OpenAPI.

> **Tip**
>
> Para comprender cómo se utilizan las extensiones de API Gateway en una aplicación, puede utilizar la consola API Gateway para crear una API REST o API HTTP y exportarla a un archivo de definición de OpenAPI. Para obtener más información sobre cómo exportar una API, consulte Exportación de una API REST desde API Gateway y Exportación de API HTTP desde API Gateway.

Temas

- Objeto x-amazon-apigateway-any-method
- Objeto x-amazon-apigateway-cors
- Propiedad x-amazon-apigateway-api-key-source
- Objeto de x-amazon-apigateway-auth
- Objeto x-amazon-apigateway-authorizer
- Propiedad x-amazon-apigateway-authtype
- Propiedad x-amazon-apigateway-binary-media-types
- Objeto x-amazon-apigateway-documentation
- x-amazon-apigateway-endpoint-access-mode
- Objeto x-amazon-apigateway-endpoint-configuration
- Objeto x-amazon-apigateway-gateway-responses
- Objeto x-amazon-apigateway-gateway-responses.gatewayResponse
- Objeto x-amazon-apigateway-gateway-responses.responseParameters
- Objeto x-amazon-apigateway-gateway-responses.responseTemplates
- x-amazon-apigateway-importexport-version
- Objeto x-amazon-apigateway-integration
- Objeto x-amazon-apigateway-integrations

- Objeto x-amazon-apigateway-integration.requestTemplates
- Objeto x-amazon-apigateway-integration.requestParameters
- Objeto x-amazon-apigateway-integration.responses
- Objeto x-amazon-apigateway-integration.response
- Objeto x-amazon-apigateway-integration.responseTemplates
- Objeto x-amazon-apigateway-integration.responseParameters
- Objeto x-amazon-apigateway-integration.tlsConfig
- x-amazon-apigateway-minimum-compression-size
- x-amazon-apigateway-policy
- Propiedad x-amazon-apigateway-request-validator
- Objeto x-amazon-apigateway-request-validators
- Objeto x-amazon-apigateway-request-validators.requestValidator
- x-amazon-apigateway-security-policy
- Propiedad x-amazon-apigateway-tag-value

Objeto x-amazon-apigateway-any-method

Especifica el objeto Operation de OpenAPI para el método catch-all `ANY` de API Gateway en un objeto Path Item de OpenAPI. Este objeto puede existir junto con otros objetos Operation y capturar cualquier método HTTP que no se haya declarado de forma explícita.

En la siguiente tabla se muestran las propiedades ampliadas por API Gateway. Para el resto de las propiedades del objeto Operation de OpenAPI, consulte la especificación de OpenAPI.

Nombre de la propiedad	Tipo	Descripción
`isDefaultRoute`	`Boolean`	Especifica si una ruta es la ruta `$default`. Solo se admite para API HTTP. Para obtener más información, consulte Creación de rutas para las API de HTTP en API Gateway.

Nombre de la propiedad	Tipo	Descripción
`x-amazon-apigateway-integration`	Objeto x-amazon-apigateway-integration	Especifica la integración del método con el backend. Se trata de una propiedad extendida del objeto Operation de OpenAPI. La integración puede ser del tipo `AWS`, `AWS_PROXY` , `HTTP`, `HTTP_PROXY` o `MOCK`.

Ejemplos de x-amazon-apigateway-any-method

En el siguiente ejemplo se integra el método `ANY` en un recurso de proxy, `{proxy+}`, con una función de Lambda, `TestSimpleProxy`.

```
    "/{proxy+}": {
      "x-amazon-apigateway-any-method": {
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "proxy",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {},
        "x-amazon-apigateway-integration": {
          "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789012:function:TestSimpleProxy/invocations",
          "httpMethod": "POST",
          "type": "aws_proxy"
        }
```

En el ejemplo siguiente se crea una ruta `$default` para una API HTTP que se integra con una función de Lambda, `HelloWorld`.

```
"/$default": {
    "x-amazon-apigateway-any-method": {
      "isDefaultRoute": true,
      "x-amazon-apigateway-integration": {
        "type": "AWS_PROXY",
        "httpMethod": "POST",
        "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789012:function:HelloWorld/invocations",
        "timeoutInMillis": 1000,
        "connectionType": "INTERNET",
        "payloadFormatVersion": 1.0
      }
   }
}
```

Objeto x-amazon-apigateway-cors

Especifica la configuración de uso compartido de recursos entre orígenes (CORS) para una API HTTP. La extensión se aplica a la estructura de OpenAPI en el nivel raíz. Para obtener más información, consulte Configuración de CORS de las API de HTTP en API Gateway.

Nombre de la propiedad	Tipo	Descripción
`allowOrigins`	`Array`	Especifica los orígenes permitidos.
`allowCredentials`	`Boolean`	Especifica si las credenciales están incluidas en la solicitud de CORS.
`exposeHeaders`	`Array`	Especifica los encabezados que se exponen.
`maxAge`	`Integer`	Especifica el número de segundos durante los que el navegador debe almacenar en caché los resultados de la solicitud preliminar.

Nombre de la propiedad	Tipo	Descripción
`allowMethods`	`Array`	Especifica los métodos HTTP permitidos.
`allowHeaders`	`Array`	Especifica los encabezados permitidos.

Ejemplo de x-amazon-apigateway-cors

A continuación, se muestra un ejemplo de configuración de CORS para una API HTTP.

```
"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"
    ]
}
```

Propiedad x-amazon-apigateway-api-key-source

Especifique el origen del que va a provenir la clave de la API para limitar los métodos de API que solicitan claves. Esta propiedad de nivel de API es de tipo `String`. Para obtener más información

sobre cómo configurar un método para exigir una clave de API, consulte the section called "Configuración de un método para usar claves de API con una definición de OpenAPI".

Especifique el origen de la clave de la API en las solicitudes. Los valores válidos son:

- `HEADER` para recibir la clave de la API desde el encabezado `X-API-Key` de una solicitud.
- `AUTHORIZER` para recibir la clave de la API de `UsageIdentifierKey` desde un autorizador de Lambda (que anteriormente se denominaba autorizador personalizado).

Ejemplo de x-amazon-apigateway-api-key-source

En el siguiente ejemplo, el encabezado `X-API-Key` se establece como origen de las claves de la API.

OpenAPI 2.0

```
{
  "swagger" : "2.0",
  "info" : {
    "title" : "Test1"
   },
  "schemes" : [ "https" ],
  "basePath" : "/import",
  "x-amazon-apigateway-api-key-source" : "HEADER",
   .
   .
   .
}
```

OpenAPI 3.0.1

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "Test1"
  },
  "servers" : [ {
    "url" : "/{basePath}",
    "variables" : {
      "basePath" : {
```

```
        "default" : "import"
      }
    }
  } ],
  "x-amazon-apigateway-api-key-source" : "HEADER",
   .
   .
   .
}
```

Objeto de x-amazon-apigateway-auth

Define un tipo de autorización que se aplicará para la autorización de invocaciones de métodos en API Gateway.

Nombre de la propiedad	Tipo	Descripción
`type`	`string`	Especifica el tipo de autorización. Especifique `"NONE"` para acceso abierto. Especifique `"AWS_IAM"` para usar permisos de IAM. En los valores, no se distingue entre mayúsculas y minúsculas.

Ejemplo de x-amazon-apigateway-auth

En el ejemplo siguiente se establece el tipo de autorización de un método de API.

OpenAPI 3.0.1

```
{
  "openapi": "3.0.1",
  "info": {
    "title": "openapi3",
    "version": "1.0"
  },
  "paths": {
    "/protected-by-iam": {
```

```
      "get": {
        "x-amazon-apigateway-auth": {
          "type": "AWS_IAM"
        }
      }
    }
  }
}
```

Objeto x-amazon-apigateway-authorizer

Define un autorizador de Lambda, el grupo de usuarios de Amazon Cognito o autorizador de JWT que se debe aplicar a la autorización de invocaciones de método en API Gateway. Esta extensión se aplica a la definición de seguridad en OpenAPI 2 y OpenAPI 3.

Nombre de la propiedad	Tipo	Descripción
`type`	`string`	El tipo del autorizador. Esta propiedad es obligatoria. Para las API REST, especifiq ue `token` para un autorizad or con la identidad del intermediario incrustada en un authorization token. Especifique `request` para un autorizador con la identidad del intermediario incluida en los parámetros de solicitud . Especifique `cognito_u ser_pools` para un autorizador que utilice un grupo de usuarios de Amazon Cognito para controlar el acceso a la API. Para API HTTP, especifique `request` para un autorizador

Nombre de la propiedad	Tipo	Descripción
		de Lambda con la identidad del intermediario incluida en los parámetros de solicitud . Especifique `jwt` para un autorizador de JWT.
`authorizerUri`	`string`	El identificador uniforme de recursos (URI) de la función de Lambda del autorizador. La sintaxis es la siguiente: `"arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/arn:aws:lambda:us-east-1:`*account-id*`:function:`*auth_function_name*`/invocations"`
`authorizerCredentials`	`string`	Las credenciales necesarias para invocar el autorizador, si existen, en forma de ARN de un rol de ejecución de IAM. Por ejemplo, "arn:aws:iam::*id-cuenta*:*rol_de_IAM* ".

Nombre de la propiedad	Tipo	Descripción
`authorizerPayloadFormatVersion`	`string`	Para las API HTTP, especifica el formato de los datos que API Gateway envía a un autorizador Lambda y cómo API Gateway interpreta la respuesta de Lambda. Para obtener más información, consulte the section called “Versión de formato de carga”.
`enableSimpleResponses`	`Boolean`	Para API HTTP, especifica si un autorizador de `request` devuelve un valor booleano o una política de IAM. Solo se admite para autorizadores con una `authorizerPayloadFormatVersion` de `2.0`. Si está habilitada, la función del autorizador de Lambda devuelve un valor booleano. Para obtener más información, consulte the section called “Respuesta de función de Lambda para el formato 2.0”.
`identitySource`	`string`	Lista separada por comas de las expresiones de mapeo de los parámetros de solicitud como en la fuente de identidad. Aplicable solo para el autorizador del tipo `request` y `jwt`.

Nombre de la propiedad	Tipo	Descripción
`jwtConfiguration`	`Object`	Especifica el emisor y los destinatarios de un autorizador de JWT. Para obtener más información, consulte JWTConfiguration en la Referencia de la API de API Gateway versión 2. Solo se admite para API HTTP.
`identityValidationExpression`	`string`	Una expresión regular para validar el token y la identidad de entrada. Por ejemplo, "^ x-[a-z]+". Compatible solo para autorizadores `TOKEN` para API de REST.
`authorizerResultTtlInSeconds`	`string`	Número de segundos durante los que se almacena en caché el resultado del autorizador.
`providerARNs`	Matriz de `string`	Una lista de los ARN de grupos de usuarios de Amazon Cognito para `COGNITO_USER_POOLS` .

Ejemplos de x-amazon-apigateway-authorizer para API REST

El siguiente ejemplo de definiciones de seguridad de OpenAPI especifica un autorizador de Lambda del tipo "token" denominado `test-authorizer`.

```
  "securityDefinitions" : {
    "test-authorizer" : {
      "type" : "apiKey",                          // Required and the value must be
 "apiKey" for an API Gateway API.
```

```
      "name" : "Authorization",                  // The name of the header containing
 the authorization token.
      "in" : "header",                           // Required and the value must be
 "header" for an API Gateway API.
      "x-amazon-apigateway-authtype" : "custom", // Specifies the authorization
 mechanism for the client.
      "x-amazon-apigateway-authorizer" : {       // An API Gateway Lambda authorizer
 definition
        "type" : "token",                        // Required property and the value
 must "token"
        "authorizerUri" : "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/
functions/arn:aws:lambda:us-east-1:account-id:function:function-name/invocations",
        "authorizerCredentials" : "arn:aws:iam::account-id:role",
        "identityValidationExpression" : "^x-[a-z]+",
        "authorizerResultTtlInSeconds" : 60
      }
    }
  }
```

El siguiente fragmento del objeto de operación de OpenAPI establece el método `GET /http` para utilizar el autorizador de Lambda precedente.

```
  "/http" : {
    "get" : {
      "responses" : { },
      "security" : [ {
        "test-authorizer" : [ ]
      } ],
      "x-amazon-apigateway-integration" : {
        "type" : "http",
        "responses" : {
          "default" : {
            "statusCode" : "200"
          }
        },
        "httpMethod" : "GET",
        "uri" : "http://api.example.com"
      }
    }
  }
```

El siguiente ejemplo de definiciones de seguridad de OpenAPI especifica un autorizador de Lambda del tipo "request" con un único parámetro de encabezado (`auth`) como fuente de identidad. El elemento `securityDefinitions` se denomina `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
      }
    }
}
```

El siguiente ejemplo de definiciones de seguridad de OpenAPI especifica un autorizador de Lambda del tipo "request" con un encabezado (`HeaderAuth1`) y un parámetro de cadena de consulta `QueryString1` como fuentes de identidad.

```
"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
      }
    }
}
```

El siguiente ejemplo de definiciones de seguridad de OpenAPI especifica un autorizador de Lambda de API Gateway del tipo "request" con una única variable de etapa (`stage`) como fuente de identidad.

```
"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
      }
    }
}
```

En el siguiente ejemplo de definición de seguridad de OpenAPI se especifica un grupo de usuarios de Amazon Cognito como autorizador.

```
"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"
       ]
     }
   }
```

En el siguiente fragmento de objeto de operación de OpenAPI se establece `GET /http` para utilizar el grupo de usuarios anterior de Amazon Cognito como autorizador, sin ámbitos personalizados.

```
  "/http" : {
     "get" : {
       "responses" : { },
       "security" : [ {
         "cognito-pool" : [ ]
       } ],
       "x-amazon-apigateway-integration" : {
         "type" : "http",
         "responses" : {
           "default" : {
             "statusCode" : "200"
           }
         },
         "httpMethod" : "GET",
         "uri" : "http://api.example.com"
       }
     }
   }
```

Ejemplos de x-amazon-apigateway-authorizer para API HTTP

En el siguiente ejemplo de OpenAPI 3.0 se crea un autorizador de JWT para una API HTTP que utiliza Amazon Cognito como proveedor de identidad, con el encabezado `Authorization` como origen de identidad.

```
"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"
    }
  }
}
```

En el siguiente ejemplo de OpenAPI 3.0 se crea el mismo autorizador de JWT que en el ejemplo anterior. Sin embargo, en este ejemplo se utiliza la propiedad `openIdConnectUrl` de OpenAPI para detectar automáticamente el emisor. La `openIdConnectUrl` debe estar completamente formada.

```
"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"
    }
  }
}
```

En el ejemplo siguiente se crea un autorizador Lambda para una API HTTP. Este autorizador de ejemplo utiliza el encabezado `Authorization` como su origen de identidad. El

autorizador utiliza la versión `2.0` del formato de carga útil y devuelve el valor booleano, ya que `enableSimpleResponses` está establecido en `true`.

```
"securitySchemes" : {
  "lambda-authorizer" : {
    "type" : "apiKey",
    "name" : "Authorization",
    "in" : "header",
    "x-amazon-apigateway-authorizer" : {
      "type" : "request",
      "identitySource" : "$request.header.Authorization",
      "authorizerUri" : "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123456789012:function:function-name/invocations",
      "authorizerPayloadFormatVersion" : "2.0",
      "authorizerResultTtlInSeconds" : 300,
      "enableSimpleResponses" : true
    }
  }
}
```

Propiedad x-amazon-apigateway-authtype

Para las API REST, esta extensión se puede utilizar a fin definir un tipo personalizado de un autorizador de Lambda. En este caso, el valor es de forma libre. Por ejemplo, una API puede tener varios autorizadores de Lambda que utilizan diferentes esquemas internos. Puede utilizar esta extensión para identificar el esquema interno de un autorizador de Lambda.

Más comúnmente, en API HTTP y API REST, también se puede usar como una forma de definir la autorización de IAM en varias operaciones que comparten el mismo esquema de seguridad. En este caso, el término `awsSigv4` es un término reservado, junto con cualquier término prefijado por `aws`.

Esta extensión se aplica al esquema de seguridad de tipo `apiKey` en OpenAPI 2 y OpenAPI 3.

Ejemplo de x-amazon-apigateway-authtype

El siguiente ejemplo de OpenAPI 3 define la autorización de IAM en varios recursos en una API REST o API HTTP:

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

El siguiente ejemplo de OpenAPI 3 define un autorizador de Lambda con un esquema personalizado para una API REST:

```
{
  "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"
}
```

Véase también

authorizer.authType

Propiedad x-amazon-apigateway-binary-media-types

Especifica la lista de tipos de medios binarios que se admiten en API Gateway, como `application/octet-stream` y `image/jpeg`. Esta extensión es una matriz JSON. Debe incluirse como una extensión de proveedor de nivel superior al documento de OpenAPI.

Ejemplo de x-amazon-apigateway-binary-media-types

El siguiente ejemplo muestra el orden de búsqueda de codificación de una API.

```
"x-amazon-apigateway-binary-media-types": [ "application/octet", "image/jpeg" ]
```

Objeto x-amazon-apigateway-documentation

Define las piezas de documentación que se van a importar a API Gateway. Este objeto es un objeto JSON que contiene una matriz de instancias `DocumentationPart`.

Nombre de la propiedad	Tipo	Descripción
`documentationParts`	`Array`	Una matriz de instancias `DocumentationPart` exportadas o importadas.
`version`	`String`	El identificador de versión de la snapshot de las piezas de documentación exportadas.

Ejemplo de x-amazon-apigateway-documentation

El siguiente ejemplo de la extensión de API Gateway para OpenAPI define instancias `DocumentationParts` para que se importen o exporten desde una API en API Gateway.

```
{ ...
  "x-amazon-apigateway-documentation": {
    "version": "1.0.3",
    "documentationParts": [
       {
         "location": {
          "type": "API"
       },
        "properties": {
          "description": "API description",
          "info": {
            "description": "API info description 4",
            "version": "API info version 3"
          }
        }
      },
      {
         … // Another DocumentationPart instance
      }
    ]
  }
}
```

x-amazon-apigateway-endpoint-access-mode

Especifica el modo de acceso de punto de conexión para una API de REST o un nombre de dominio personalizado. Para obtener más información sobre el modo de acceso de punto de conexión, consulte Modo de acceso de punto de conexión.

`x-amazon-apigateway-endpoint-access-mode`Ejemplo de

En el ejemplo siguiente se especifica el modo de acceso de punto de conexión estricto.

```
"x-amazon-apigateway-endpoint-access-mode": "STRICT"
```

Objeto x-amazon-apigateway-endpoint-configuration

Especifica detalles de la configuración del punto de enlace para una API. Esta extensión es una propiedad extendida del objeto Operation de OpenAPI. Este objeto debe estar presente en las

extensiones de proveedor de nivel superior de Swagger 2.0. Para OpenAPI 3.0, debe estar presente en las extensiones de proveedor del objeto Server.

Nombre de la propiedad	Tipo	Descripción
`disableExecuteApiEndpoint`	Booleano	Especifica si los clientes pueden invocar la API mediante el punto de enlace `execute-api` predeterminado. De forma predeterminada, los clientes pueden invocar su API con el punto de enlace `https://{api_id}.execute-api.{region}.amazonaws.com` predeterminado. Para exigir que los clientes utilicen un nombre de dominio personalizado para invocar su API, especifique `true`.
`vpcEndpointIds`	Matriz de `String`	Una lista de identificadores de VpcEndpoint en la que crear registros de alias de Route 53 para una API REST. Solo es compatible con las API REST del tipo de punto de enlace `PRIVATE`.
`ipAddressType`	`string`	Los tipos de direcciones IP que pueden invocar una API HTTP. Utilice `ipv4` para permitir que los tipos de direcciones IPv4 invoquen una API HTTP. Utilice `dualstack` para permitir que los tipos de direcciones IPv4 e IPv6

Nombre de la propiedad	Tipo	Descripción
		invoquen una API HTTP. Solo se admite para las API HTTP.

Ejemplos de x-amazon-apigateway-endpoint-configuration

En el siguiente ejemplo se asocian puntos de enlace de la VPC especificados a la API REST.

```
"x-amazon-apigateway-endpoint-configuration": {
    "vpcEndpointIds": ["vpce-0212a4ababd5b8c3e", "vpce-01d622316a7df47f9"]
}
```

En el ejemplo siguiente se desactiva el punto de enlace predeterminado de una API.

```
"x-amazon-apigateway-endpoint-configuration": {
    "disableExecuteApiEndpoint": true
}
```

El siguiente ejemplo establece el tipo de dirección IP en pila doble para una API HTTP.

```
"x-amazon-apigateway-endpoint-configuration": {
    "ipAddressType": "dualstack"
}
```

Objeto x-amazon-apigateway-gateway-responses

Define las respuestas de gateway de una API como un mapa de pares de clave-valor de cadena a GatewayResponse. La extensión se aplica a la estructura de OpenAPI en el nivel raíz.

Nombre de la propiedad	Tipo	Descripción
responseType	x-amazon-apigateway-gateway-responses.gatewayResponse	Un atributo `GatewayResponse` para el elemento *responseType* especificado.

Ejemplo de x-amazon-apigateway-gateway-responses

El siguiente ejemplo de extensión de API Gateway para OpenAPI define un mapa de GatewayResponses que contiene dos instancias de GatewayResponse, una para el tipo `DEFAULT_4XX` y otra para el tipo `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 }"
      }
    }
  }
}
```

Objeto x-amazon-apigateway-gateway-responses.gatewayResponse

Define una respuesta de gateway de un tipo de respuesta determinado, incluido el código de estado, todos los parámetros de respuesta aplicables o las plantillas de respuesta.

Nombre de la propiedad	Tipo	Descripción
responseParameters	x-amazon-apigateway-gateway-responses.responseParameters	Especifica los parámetros de GatewayResponse, es decir, los parámetros de encabezado. Los valores de los parámetros pueden tomar cualquier valor de parámetro

Nombre de la propiedad	Tipo	Descripción
		de solicitud entrante o un valor personalizado estático.
responseTemplates	x-amazon-apigateway-gateway-responses.responseTemplates	Especifica las plantillas de mapeo de la respuesta de gateway. El motor de VTL no procesa las plantillas.
statusCode	`string`	Un código de estado HTTP para la respuesta de gateway.

Ejemplo de x-amazon-apigateway-gateway-responses.gatewayResponse

El siguiente ejemplo de extensión de API Gateway para OpenAPI define un objeto GatewayResponse para personalizar la respuesta `INVALID_API_KEY` y devolver el código de estado `456`, el valor del encabezado `api-key` de la solicitud entrante y un mensaje `"Bad api-key"`.

```
"INVALID_API_KEY": {
  "statusCode": "456",
  "responseParameters": {
    "gatewayresponse.header.api-key": "method.request.header.api-key"
  },
  "responseTemplates": {
    "application/json": "{\"message\": \"Bad api-key\" }"
  }
}
```

Objeto x-amazon-apigateway-gateway-responses.responseParameters

Define un mapa de cadena a cadena de pares de clave-valor para generar parámetros de respuesta de gateway a partir de los parámetros de la solicitud entrante o mediante cadenas literales. Compatible solo con API REST.

Nombre de la propiedad	Tipo	Descripción
`gatewayresponse. param-position .param-name`	`string`	*`param-position`* puede ser `header`, `path` o `querystring` . Para obtener más información, consulte Asignación de parámetros para las API de REST en API Gateway.

Ejemplo de x-amazon-apigateway-gateway-responses.responseParameters

El siguiente ejemplo de extensión de OpenAPI muestra una expresión de asignación de parámetros de respuesta GatewayResponse para habilitar la compatibilidad de CORS con recursos en los dominios `*.example.domain`.

```
"responseParameters": {
  "gatewayresponse.header.Access-Control-Allow-Origin": '*.example.domain',
  "gatewayresponse.header.from-request-header" : method.request.header.Accept,
  "gatewayresponse.header.from-request-path" : method.request.path.petId,
  "gatewayresponse.header.from-request-query" : method.request.querystring.qname
}
```

Objeto x-amazon-apigateway-gateway-responses.responseTemplates

Define las plantillas de asignación de GatewayResponse, como un mapa de cadena a cadena de pares de clave-valor, para una respuesta de gateway especificada. En cada par clave-valor, la clave es el tipo de contenido. Por ejemplo, "application/json" y el valor es una plantilla de mapeo stringified para sustituciones de variables simples. El motor de Velocity Template Language (VTL) no procesa una plantilla de mapeo de `GatewayResponse`.

Nombre de la propiedad	Tipo	Descripción
content-type	string	Una plantilla de mapeo de cuerpo de GatewayResponse que admite únicamente sustituciones de variables sencillas para personalizar un cuerpo de respuesta de gateway.

Ejemplo de x-amazon-apigateway-gateway-responses.responseTemplates

El siguiente ejemplo de extensión de OpenAPI muestra una plantilla de asignación de GatewayResponse para personalizar una respuesta de error generada por API Gateway en un formato específico de la aplicación.

```
      "responseTemplates": {
        "application/json": "{ \"message\": $context.error.messageString, \"type\": 
$context.error.responseType, \"statusCode\": '488' }"
      }
```

El siguiente ejemplo de extensión de OpenAPI muestra una plantilla de asignación de GatewayResponse para invalidar una respuesta de error generada por API Gateway con un mensaje de error estático.

```
      "responseTemplates": {
        "application/json": "{ \"message\": 'API-specific errors' }"
      }
```

x-amazon-apigateway-importexport-version

Especifica la versión del algoritmo de importación y exportación de API Gateway para las API HTTP. Actualmente el único valor admitido es `1.0`. Para obtener más información, consulte exportVersion en la Referencia de la API de la API Gateway versión 2.

Ejemplo de x-amazon-apigateway-importexport-version

En el ejemplo siguiente se establece la versión de importación y exportación en `1.0`.

```
{
    "openapi": "3.0.1",
    "x-amazon-apigateway-importexport-version": "1.0",
    "info": { ...
```

Objeto x-amazon-apigateway-integration

Especifica los detalles de la integración del backend utilizada para este método. Esta extensión es una propiedad extendida del objeto Operation de OpenAPI. El resultado es un objeto de integración de API Gateway.

Nombre de la propiedad	Tipo	Descripción
`cacheKeyParameters`	Matriz de `string`	Una lista de parámetros de la solicitud cuyos valores se van a almacenar en caché.
`cacheNamespace`	`string`	Un grupo de etiquetas específicas de la API de parámetros almacenados en caché relacionados.
`connectionId`	`string`	ID de un enlace VpcLink de la integración privada.
`connectionType`	`string`	Tipo de conexión de la integración. El valor válido es `"VPC_LINK"` para la integración privada o `"INTERNET"` para los demás casos.
`credentials`	`string`	Para las credenciales basadas en roles de AWS IAM,

Nombre de la propiedad	Tipo	Descripción
		especifique el ARN de un rol de IAM. Si no se especific a, las credenciales adoptan de manera predeterminada los permisos basados en recursos que se deben añadir manualmente para permitir a la API tener acceso al recurso. Para obtener más informaci ón, consulte Concesión de permisos mediante una política de recursos. Nota: Cuando utilice credencia les de IAM, asegúrese de que los puntos de enlace regionales STS de AWS están habilitados para la región en la que se implementa esta API para obtener un rendimiento óptimo.

Nombre de la propiedad	Tipo	Descripción
`contentHandling`	`string`	Tipos de conversión de codificación de la carga de solicitud. Los valores válidos son 1) `CONVERT_TO_TEXT` , para convertir una carga binaria en una cadena codificada en base64 o para convertir una carga de texto en una cadena codificada en `utf-8` o para transferir la carga de texto de forma nativa sin modificaciones, y 2) `CONVERT_TO_BINARY` , para convertir una carga de texto en un blob descodificado en base64 o para transferir una carga binaria de forma nativa sin modificaciones.
`httpMethod`	`string`	El método HTTP utilizado en la solicitud de integración. Para las invocaciones de funciones de Lambda, el valor debe ser `POST`.
`integrationSubtype`	`string`	Especifica el subtipo de integración para una integración del servicio de AWS. Solo se admite para API HTTP. Para obtener información sobre los subtipos de integración compatibles, consulte the section called “AWSReferencia de integraciones de servicios de ”.

Nombre de la propiedad	Tipo	Descripción
`integrationTarget`	`string`	El oyente de ALB o NLB al que se va a enviar la solicitud . Compatible solo con integraciones privadas que utilizan enlaces de VPC V2. Para obtener más información, consulte the section called “Enlaces de VPC V2”.
`passthroughBehavior`	`string`	Especifica cómo se pasa una carga de solicitud de tipo de contenido sin asignar a través de la solicitud de integración sin modificación. Los valores admitidos son `when_no_templates` , `when_no_match` y `never`. Para obtener más información, consulte Integration.passthroughBehavior.

Nombre de la propiedad	Tipo	Descripción
`payloadFormatVersion`	`string`	Especifica el formato de la carga enviada a una integración. Es necesario para las API HTTP. Para las API HTTP, los valores admitidos para las integraciones de proxy Lambda son `1.0` y `2.0`. Para el resto de integraciones, `1.0` es el único valor admitido. Para obtener más información, consulte the section called “AWS LambdaIntegraciones de ” y the section called “AWSReferencia de integraciones de servicios de ”.

Nombre de la propiedad	Tipo	Descripción
`requestParameters`	Objeto x-amazon-apigateway-integration.requestParameters	Para API REST, especific a las asignaciones de los parámetros de solicitud de método a los parámetros de solicitud de integración. Los parámetros de solicitud admitidos son `querystri ng` , `path`, `header` y `body`. Para las API HTTP, los parámetros de solicitud son un mapeo de clave-valor que especifica los parámetros que se pasan a integraciones de `AWS_PROXY` con `integrati onSubtype` especificado. Puede proporcionar valores estáticos o datos de solicitud de mapeo, variables de etapa o variables de contexto que se evalúan en tiempo de ejecución. Para obtener más información, consulte the section called “AWSInteg raciones de los servicios de ”.
`requestTemplates`	Objeto x-amazon-apigateway-integration.requestTemplates	Plantillas de mapeo para una carga de solicitud de tipos MIME especificados.

Nombre de la propiedad	Tipo	Descripción
`responses`	Objeto x-amazon-apigateway-integration.responses	Define las respuestas del método y especifica las asignaciones de parámetros o de carga deseadas desde respuestas de integración a respuestas de método.
`responseTransferMode`	`string`	El modo de transferencia de respuesta de la integración. Use `BUFFERED` para que API Gateway espere a recibir la respuesta completa antes de iniciar la transmisión. Use `STREAM` para que API Gateway envíe de vuelta respuestas parciales al cliente a medida que estén disponibles. Para obtener más información, consulte the section called “Transmisión de la respuesta de integración para las integraciones de proxy”.
`timeoutInMillis`	`integer`	Tiempos de espera de la integración comprendidos entre 50 ms y 29.000 ms.

Nombre de la propiedad	Tipo	Descripción
`type`	`string`	El tipo de integración con el backend especificado. Los valores válidos son: • `http` o `http_proxy` : para la integración con un backend HTTP. • `aws_proxy` : para la integración con las funciones de AWS Lambda. • `aws`: para la integración con las funciones de AWS Lambda u otros servicios de AWS, como Amazon DynamoDB, Amazon Simple Notification Service o Amazon Simple Queue Service. • `mock`, para la integración con API Gateway sin invocar ningún backend. Para obtener más información sobre los tipos de integración, consulte integration:type.
`tlsConfig`	the section called “x-amazon-apigateway-integration.tlsConfig”	Especifica la configuración de TLS para una integración.

Nombre de la propiedad	Tipo	Descripción
`uri`	`string`	El URI del punto de enlace del backend. Para integraciones del tipo `aws`, este es un valor de ARN. Para la integración HTTP, esta es la dirección URL del punto de enlace HTTP incluido el esquema `https` o `http`.

Ejemplos de x-amazon-apigateway-integration

En API HTTP, puede definir integraciones en la sección de componentes de la definición de OpenAPI. Para obtener más información, consulte Objeto x-amazon-apigateway-integrations.

```
"x-amazon-apigateway-integration": {
    "$ref": "#/components/x-amazon-apigateway-integrations/integration1"
}
```

El siguiente ejemplo se crea una integración con una función de Lambda. Para fines de demostración, en las plantillas de asignación de ejemplo en `requestTemplates` y `responseTemplates` de los ejemplos que aparecen a continuación se asume que se aplica la siguiente carga con formato JSON: `{ "name":"value_1", "key":"value_2", "redirect": {"url" :"..."} }` para generar una salida JSON `{ "stage":"value_1", "user-id":"value_2" }` o una salida XML `<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'"
            }
        }
    }
}
```

Tenga en cuenta que las comillas dobles (") de la cadena JSON en las plantillas de mapeo deben incluirse en una secuencia de escape (\").

Objeto x-amazon-apigateway-integrations

Define una colección de integraciones. Puede definir integraciones en la sección de componentes de la definición de OpenAPI y reutilizar las integraciones para varias rutas. Solo se admite para API HTTP.

Nombre de la propiedad	Tipo	Descripción
integration	Objeto x-amazon-apigateway-integration	Una colección de objetos de integración.

Ejemplo de x-amazon-apigateway-integrations

En el ejemplo siguiente se crea una API HTTP que define dos integraciones y se hace referencia a las integraciones mediante `$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"
            }
        }
    }
}
```

Objeto x-amazon-apigateway-integration.requestTemplates

Especifica las plantillas de mapeo de una carga de solicitud de los tipos MIME especificados.

Nombre de la propiedad	Tipo	Descripción
MIME type	`string`	Un ejemplo de tipo MIME es `application/json` . Para obtener información acerca de la creación de una plantilla de mapeo, consulte the section called “Transformaciones de plantillas de asignación”.

Ejemplo de x-amazon-apigateway-integration.requestTemplates

El siguiente ejemplo especifica las plantillas de mapeo de una carga de solicitud de los tipos MIME `application/json` y `application/xml`.

```
"requestTemplates" : {
    "application/json" : "#set ($root=$input.path('$')) { \"stage\": \"$root.name\",
 \"user-id\": \"$root.key\" }",
    "application/xml" : "#set ($root=$input.path('$')) <stage>$root.name</stage> "
```

```
}
```

Objeto x-amazon-apigateway-integration.requestParameters

Para API REST, especifica las asignaciones de los parámetros de solicitud de método designados a los parámetros de solicitud de integración. Los parámetros de solicitud de método deben estar definidos para poder hacer referencia a ellos.

Para API HTTP, especifica los parámetros que se pasan a integraciones de `AWS_PROXY` con un `integrationSubtype` especificado.

Nombre de la propiedad	Tipo	Descripción
`integration.request.` *`<param-type>`* `.` *`<param-name>`*	`string`	Para API REST, el valor es típicamente un parámetro de solicitud de método predefinido con el formato `method.request.` *`<param-type>`* `.` *`<param-name>`* , donde `<param-type>` puede ser `querystring` , `path`, `header` o `body`. Sin embargo, `$context.` *`VARIABLE_NAME`* , `$stageVariables.` *`VARIABLE_NAME`* , y *`STATIC_VALUE`* son también válidas. Para el parámetro `body`, `<param-name>` es una expresión de ruta JSON sin el prefijo `$.`
`parameter`	`string`	Para las API HTTP, los parámetros de solicitud son un mapeo de clave-valor que especifica los parámetros que

Nombre de la propiedad	Tipo	Descripción
		se pasan a integraciones de `AWS_PROXY` con `integrationSubtype` especificado. Puede proporcionar valores estáticos o datos de solicitud de mapeo, variables de etapa o variables de contexto que se evalúan en tiempo de ejecución. Para obtener más información, consulte the section called “AWSIntegraciones de los servicios de ”.

`x-amazon-apigateway-integration.requestParameters`Ejemplo de

El siguiente ejemplo de asignaciones de parámetros de solicitud traduce la consulta (`version`), el encabezado, (`x-user-id`) y los parámetros de ruta (`service`) de la solicitud de método en la consulta (`stage`), encabezado (`x-userid`) y los parámetros de ruta (op) de la solicitud de integración, respectivamente.

> **Note**
>
> Si está creando recursos a través de OpenAPI o CloudFormation, los valores estáticos deben escribirse entre comillas simples.
> Para añadir este valor desde la consola, escriba `application/json` en el cuadro, sin las comillas.

```
"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"
},
```

Objeto x-amazon-apigateway-integration.responses

Define las respuestas del método y especifica las asignaciones de parámetros o de carga desde respuestas de integración a respuestas de método.

Nombre de la propiedad	Tipo	Descripción
Patrón de estado de respuesta	Objeto x-amazon-apigateway-integration.response	Ya sea una expresión regular utilizada para hacer coincidir la respuesta de integración con la respuesta del método, o `default` para detectar cualquier respuesta que no haya configurado. Para las integraciones HTTP, la RegEx se aplica al código de estado de la respuesta de integración. Para las invocaciones de Lambda, la expresión regular se aplica al campo `errorMessage` del objeto de información del error que devuelve AWS Lambda como un cuerpo de respuesta de error cuando la ejecución de la función de Lambda produce una excepción. **Note** El nombre de propiedad *Patrón de estado de*

Nombre de la propiedad	Tipo	Descripción
		respuesta hace referencia a un código de estado de respuesta o a una expresión regular que describe un grupo de códigos de estado de respuesta. No se corresponde con ningún identificador de un recurso IntegrationResponse en la API REST de API Gateway.

`x-amazon-apigateway-integration.responses`Ejemplo de

El siguiente ejemplo muestra una lista de respuestas `2xx` y `302`. Para la respuesta `2xx`, la respuesta del método se asigna desde la carga de la respuesta de integración del tipo MIME `application/json` o `application/xml`. Esta respuesta utiliza las plantillas de mapeo proporcionadas. Para la respuesta `302`, la respuesta del método devuelve un encabezado `Location` cuyo valor se obtiene de la propiedad `redirect.url` de la carga de la respuesta de integración.

```
"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"
        }
    }
}
```

Objeto x-amazon-apigateway-integration.response

Define una respuesta y especifica asignaciones de parámetros o asignaciones de carga desde la respuesta de integración a la respuesta del método.

Nombre de la propiedad	Tipo	Descripción
`statusCode`	`string`	Código de estado HTTP para la respuesta de método (por ejemplo,, `"200"`. Se debe corresponder con una respuesta coincidente en el campo OpenAPI Operation `responses` .
`responseTemplates`	Objeto x-amazon-apigateway-integration.responseTemplates	Especifica las plantillas de mapeo específicas del tipo MIME para la carga de la respuesta.
`responseParameters`	Objeto x-amazon-apigateway-integration.responseParameters	Especifica las asignaciones de parámetros para la respuesta . Solo los parámetros `header` y `body` de la respuesta de integración se pueden asignar a los parámetros `header` del método.
`contentHandling`	`string`	Tipos de conversión de codificación de la carga de

Nombre de la propiedad	Tipo	Descripción
		la respuesta. Los valores válidos son 1) `CONVERT_TO_TEXT` , para convertir una carga binaria en una cadena codificada en base64 o para convertir una carga de texto en una cadena codificada en `utf-8` o para transferir la carga de texto de forma nativa sin modificaciones, y 2) `CONVERT_TO_BINARY` , para convertir una carga de texto en un blob descodificado en base64 o para transferir una carga binaria de forma nativa sin modificaciones.

`x-amazon-apigateway-integration.response`Ejemplo de

El siguiente ejemplo define una respuesta `302` para el método que obtiene una carga del tipo MIME `application/json` o `application/xml` del backend. La respuesta utiliza las plantillas de mapeo proporcionadas y devuelve la URL de redireccionamiento de la respuesta de integración en el encabezado `Location` del método.

```
{
    "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"
    }
}
```

Objeto x-amazon-apigateway-integration.responseTemplates

Especifica las plantillas de mapeo de una carga de respuesta de los tipos MIME especificados.

Nombre de la propiedad	Tipo	Descripción
MIME type	`string`	Especifica una plantilla de mapeo para transformar el cuerpo de la respuesta de integración en el cuerpo de la respuesta del método para un tipo MIME especificado. Para obtener información acerca de la creación de una plantilla de mapeo, consulte the section called “Transformaciones de plantillas de asignación”. Un ejemplo de *tipo MIME* es `application/json` .

Ejemplo de x-amazon-apigateway-integration.responseTemplate

El siguiente ejemplo especifica las plantillas de mapeo de una carga de solicitud de los tipos MIME `application/json` y `application/xml`.

```
"responseTemplates" : {
    "application/json" : "#set ($root=$input.path('$')) { \"stage\": \"$root.name\",
 \"user-id\": \"$root.key\" }",
    "application/xml" : "#set ($root=$input.path('$')) <stage>$root.name</stage> "
}
```

Objeto x-amazon-apigateway-integration.responseParameters

Especifica las asignaciones de parámetros del método de integración a parámetros de la respuesta del método. Puede mapear `header`, `body` o valores estáticos al tipo `header` de la respuesta del método. Compatible solo con API REST.

Nombre de la propiedad	Tipo	Descripción
`method.response.header. <param-name>`	`string`	El valor del parámetro designado se puede obtener a partir de los tipos `header` y `body` de los parámetros de la respuesta de integración.

`x-amazon-apigateway-integration.responseParameters`Ejemplo de

El siguiente ejemplo asigna los parámetros `body` y `header` de la respuesta de integración a dos parámetros `header` de la respuesta del método.

```
"responseParameters" : {
    "method.response.header.Location" : "integration.response.body.redirect.url",
    "method.response.header.x-user-id" : "integration.response.header.x-userid"
}
```

Objeto x-amazon-apigateway-integration.tlsConfig

Especifica la configuración de TLS para una integración.

Nombre de la propiedad	Tipo	Descripción
`insecureSkipVerification`	`Boolean`	Compatible solo con API REST. Especifica si API Gateway omite o no la

Nombre de la propiedad	Tipo	Descripción
		verificación de que el certificado para un punto de enlace de integración es emitido por una entidad de certificación admitida. No se recomienda, pero le permite utilizar certificados firmados por entidades de certificación privadas o certificados autofirmados. Si está habilitado, API Gateway sigue realizando la validación básica del certificado, que incluye comprobar la fecha de vencimiento del certificado, el nombre de host, la presencia de una autoridad de certificación raíz y la integridad de la cadena de certificados. La validación de la cadena de certificados incluye la aplicación de las restricciones de nombre X509v3. Si se producen errores de validación con la opción `insecureSkipVerification` habilitada, compruebe que los certificados CA no incluyan extensiones de restricciones de nombres o vuelva a emitirlos sin esa extensión. El certificado raíz que pertenece a la autoridad privada debe cumplir las siguientes restricciones:

Nombre de la propiedad	Tipo	Descripción
		• La extensión x509 `keyUsage` debe tener `keyCertSign` . • La extensión x509 `basicConstraints` debe tener `CA:TRUE`. Compatible solo para integraciones de `HTTP` y `HTTP_PROXY` . **Warning** No se recomienda habilitar `insecureSkipVerification` , especialmente para integraciones con puntos de conexión HTTPS públicos. Si habilita `insecureSkipVerification` , aumenta el riesgo de ataques MITM (man-in-the-middle).

Nombre de la propiedad	Tipo	Descripción
`serverNameToVerify`	`string`	Compatible únicamente con integraciones privadas de API HTTP. Si especific a un nombre de servidor, API Gateway lo utiliza para verificar el nombre de host en el certificado de integración. El nombre del servidor también se incluye en el protocolo de enlace TLS para admitir la indicación de nombre del servidor (SNI) o el alojamiento virtual.

Ejemplos de x-amazon-apigateway-integration.tlsConfig

En el siguiente ejemplo de OpenAPI 3.0 se habilita `insecureSkipVerification` una integración de proxy HTTP de API REST.

```
"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",
}
```

En el siguiente ejemplo de OpenAPI 3.0 se especifica un `serverNameToVerify` para una integración privada de API HTTP.

```
"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-compression-size

Especifica el tamaño mínimo de compresión de una API REST. Para habilitar la compresión, especifique un número entero entre 0 y 10485760. Para obtener más información, consulte Compresión de la carga útil para las API de REST en API Gateway.

Ejemplo de x-amazon-apigateway-minimum-compression-size

En el siguiente ejemplo se especifica un tamaño mínimo de compresión de `5242880` bytes para una API REST.

```
"x-amazon-apigateway-minimum-compression-size": 5242880
```

x-amazon-apigateway-policy

Especifica una política de recursos para una API REST. Para obtener más información sobre las políticas de recursos, consulte Control del acceso a una API de REST con políticas de recursos de API Gateway. Para obtener ejemplos de políticas de recursos, consulte Ejemplos de políticas de recursos de API Gateway.

`x-amazon-apigateway-policy`Ejemplo de

En el ejemplo siguiente se especifica una política de recursos para una API REST. La política de recursos deniega (bloquea) el tráfico entrante a una API desde un bloque de dirección IP de origen especificado. En la importación, `"execute-api:/*"` se convierte en `arn:aws:execute-`

`api:region:account-id:api-id/*`, y se utiliza la región actual, el ID de la cuenta de AWS y el ID actual de la API REST.

```
"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"
                }
            }
        }
    ]
}
```

Propiedad x-amazon-apigateway-request-validator

Especifica un validador de solicitudes, haciendo referencia a un *request_validator_name* del mapa Objeto x-amazon-apigateway-request-validators, para habilitar la validación de solicitudes en la API contenedora o en un método. El valor de esta extensión es una cadena JSON.

Esta extensión puede especificarse en el nivel de API o en el nivel de método. El validador de nivel de API se aplica a todos los métodos, a menos que lo invalide el validador de nivel de método.

`x-amazon-apigateway-request-validator`Ejemplo de

El siguiente ejemplo aplica el validador de solicitudes `basic` en el nivel de API aplicando al mismo tiempo el validador de solicitudes `parameter-only` a la solicitud `POST /validation`.

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "x-amazon-apigateway-request-validators" : {
    "basic" : {
      "validateRequestBody" : true,
      "validateRequestParameters" : true
    },
    "params-only" : {
      "validateRequestBody" : false,
      "validateRequestParameters" : true
    }
  },
  "x-amazon-apigateway-request-validator" : "basic",
  "paths": {
    "/validation": {
      "post": {
        "x-amazon-apigateway-request-validator" : "params-only",
        ...
     }
}
```

Objeto x-amazon-apigateway-request-validators

Define los validadores de solicitudes admitidos para la API contenedora como un mapa entre un nombre de validador y las reglas de validación de solicitudes asociadas. Esta extensión se aplica a una API REST

Nombre de la propiedad	Tipo	Descripción
`request_validator_name`	Objeto x-amazon-apigateway-request-validators.requestValidator	Especifica las reglas de validación del validador designado. Por ejemplo:

Nombre de la propiedad	Tipo	Descripción

		<pre>"basic" : {
		"validate
		RequestBody" : true,
		"validate
		RequestParameters" :
		true
		},</pre> Para aplicar este validador a un método específico, haga referencia al nombre del validador (`basic`) como el valor de la propiedad Propiedad x-amazon-apigateway-request-validator.

`x-amazon-apigateway-request-validators`Ejemplo de

El siguiente ejemplo muestra un conjunto de validadores de solicitudes para una API como un mapa entre un nombre de validador y las reglas de validación de solicitudes asociadas.

OpenAPI 2.0

```
{
  "swagger": "2.0",
  ...
  "x-amazon-apigateway-request-validators" : {
    "basic" : {
      "validateRequestBody" : true,
      "validateRequestParameters" : true
    },
    "params-only" : {
      "validateRequestBody" : false,
      "validateRequestParameters" : true
    }
  },
  ...
```

```
}
```

Objeto x-amazon-apigateway-request-validators.requestValidator

Especifica las reglas de validación de un validador de solicitudes como parte de la definición del mapa Objeto x-amazon-apigateway-request-validators.

Nombre de la propiedad	Tipo	Descripción
`validateRequestBody`	`Boolean`	Especifica si se debe validar el cuerpo de la solicitud (`true`) o no (`false`).
`validateRequestParameters`	`Boolean`	Especifica si se deben validar los parámetros de solicitud obligatorios (`true`) o no (`false`).

`x-amazon-apigateway-request-validators.requestValidator`Ejemplo de

El siguiente ejemplo muestra un validador de solicitudes de solo parámetros:

```
"params-only": {
    "validateRequestBody" : false,
    "validateRequestParameters" : true
}
```

x-amazon-apigateway-security-policy

Especifica una política de seguridad para una API de REST. Si crea una política de seguridad que comience por "`SecurityPolicy_`", también debe configurar el modo de acceso de punto de conexión. Para obtener más información sobre las políticas de seguridad, consulte Políticas de seguridad de las API de REST en API Gateway.

`x-amazon-apigateway-security-policy`Ejemplo de

En el ejemplo siguiente se especifica `SecurityPolicy_TLS13_1_3_2025_09` para una API de REST.

```
"x-amazon-apigateway-security-policy": "SecurityPolicy_TLS13_1_3_2025_09"
```

Propiedad x-amazon-apigateway-tag-value

Especifica el valor de una etiqueta de AWS para una API HTTP. Puede utilizar la propiedad `x-amazon-apigateway-tag-value` como parte del objeto de etiqueta OpenAPI de nivel de raíz para especificar etiquetas de AWS para una API HTTP. Si especifica un nombre de etiqueta sin la propiedad `x-amazon-apigateway-tag-value`, API Gateway crea una etiqueta con una cadena vacía para un valor.

Para obtener más información acerca del etiquetado, consulte Etiquetado de recursos de API Gateway.

Nombre de la propiedad	Tipo	Descripción
`name`	`String`	Especifica la clave de etiqueta.
`x-amazon-apigateway-tag-value`	`String`	Especifica el valor de la etiqueta.

`x-amazon-apigateway-tag-value`Ejemplo de

En el ejemplo siguiente se especifican dos etiquetas para una API HTTP:

- "Owner": "Admin"
- "Prod": ""

```
"tags": [
    {
      "name": "Owner",
```

```
      "x-amazon-apigateway-tag-value": "Admin"
    },
    {
      "name": "Prod"
    }
]
```

Seguridad en Amazon API Gateway

La seguridad en la nube de AWS es la máxima prioridad. Como cliente de AWS, se beneficia de una arquitectura de red y un centro de datos que se han diseñado para satisfacer los requisitos de seguridad de las organizaciones más exigentes.

La seguridad es una responsabilidad compartida entre AWS y usted. El modelo de responsabilidad compartida la describe como seguridad de la nube y seguridad en la nube:

- Seguridad de la nube: AWS es responsable de proteger la infraestructura que ejecuta los servicios de AWS en la nube de AWS. AWS también proporciona servicios que puede utilizar de forma segura. Los auditores externos prueban y verifican periódicamente la eficacia de nuestra seguridad como parte de los AWSProgramas de conformidad de . Para obtener información sobre los programas de conformidad que se aplican a Amazon API Gateway, consulte los Servicios de AWS en el ámbito del programa de conformidad.
- Seguridad en la nube: su responsabilidad viene determinada por el servicio de AWS que utilice. También es responsable de otros factores, incluida la confidencialidad de los datos, los requisitos de la empresa y la legislación y la normativa aplicables.

Esta documentación le ayuda a comprender cómo puede aplicar el modelo de responsabilidad compartida cuando se utiliza API Gateway. En los siguientes temas, se le mostrará cómo configurar API Gateway para satisfacer sus objetivos de seguridad y conformidad. También aprenderá a utilizar otros servicios de AWS que le ayudarán a supervisar y a proteger los recursos de API Gateway.

Para obtener más información, consulte Información general sobre seguridad de Amazon API Gateway.

Temas

- Protección de datos en Amazon API Gateway
- Identity and Access Management para Amazon API Gateway
- Registro y monitoreo en Amazon API Gateway
- Validación de conformidad para Amazon API Gateway
- Resiliencia en Amazon API Gateway
- Seguridad de la infraestructura en Amazon API Gateway
- Análisis de vulnerabilidades en Amazon API Gateway

- Prácticas recomendadas sobre seguridad para Amazon API Gateway

Protección de datos en Amazon API Gateway

El modelo de responsabilidad compartida de AWS se aplica a la protección de datos en Amazon API Gateway. Como se describe en este modelo, AWS es responsable de proteger la infraestructura global que ejecuta toda la Nube de AWS. Eres responsable de mantener el control sobre el contenido alojado en esta infraestructura. También eres responsable de las tareas de administración y configuración de seguridad para los Servicios de AWS que utiliza. Para obtener más información sobre la privacidad de datos, consulte Preguntas frecuentes sobre la privacidad de datos. Para obtener más información sobre la protección de datos en Europa, consulte el Centro del Reglamento General de Protección de Datos (RGPD).

Con fines de protección de datos, recomendamos proteger las credenciales de la Cuenta de AWS y configurar cuentas de usuario individuales con AWS IAM Identity Center o AWS Identity and Access Management (IAM). De esta manera, solo se otorgan a cada usuario los permisos necesarios para cumplir sus obligaciones laborales. También recomendamos proteger sus datos de la siguiente manera:

- Utiliza la autenticación multifactor (MFA) en cada cuenta.
- Utiliza SSL/TLS para comunicarse con los recursos de AWS. Exigimos TLS 1.2 y recomendamos TLS 1.3.
- Configure los registros de API y de actividad de los usuarios con AWS CloudTrail. Para obtener información sobre cómo utilizar registros de seguimiento de CloudTrail para capturar actividades de AWS, consulte Working with CloudTrail trails en la Guía del usuario de AWS CloudTrail.
- Utiliza las soluciones de cifrado de AWS, junto con todos los controles de seguridad predeterminados dentro de los servicios de Servicios de AWS.
- Utiliza servicios de seguridad administrados avanzados, como Amazon Macie, que lo ayuden a detectar y proteger la información confidencial almacenada en Amazon S3.
- Si necesita módulos criptográficos validados FIPS 140-3 al acceder a AWS a través de una interfaz de la línea de comandos o una API, utiliza un punto de conexión de FIPS. Para obtener más información sobre los puntos de conexión de FIPS disponibles, consulte Estándar de procesamiento de la información federal (FIPS) 140-3.

Se recomienda encarecidamente no introducir nunca información confidencial o sensible, como por ejemplo, direcciones de correo electrónico de clientes, en etiquetas o campos de formato libre,

tales como el campo Nombre. Esto incluye cuando trabaje con API Gateway u otros Servicios de AWS a través de la consola, la API, la AWS CLI o los AWS SDK. Cualquier dato que introduzca en etiquetas o campos de formato libre utilizados para los nombres se pueden emplear para los registros de facturación o diagnóstico. Si proporciona una URL a un servidor externo, recomendamos encarecidamente que no incluya información de credenciales en la URL a fin de validar la solicitud para ese servidor.

Cifrado de datos en Amazon API Gateway

La protección de datos consiste en proteger los datos mientras están en tránsito (cuando viajan a y desde API Gateway) y en reposo (mientras están almacenados en AWS).

Cifrado de datos en reposo en Amazon API Gateway

Si elige habilitar el almacenamiento en caché para una API de REST, puede habilitar el cifrado en caché. Para obtener más información, consulte Configuración de caché para las API de REST en API Gateway.

Para obtener más información sobre la protección de datos, consulte la entrada de blog relativa al modelo de responsabilidad compartida de AWS y GDPR en el blog de seguridad de AWS.

Cifrado y descifrado de la clave privada de su certificado

Cuando crea un nombre de dominio personalizado para API privadas, su certificado de ACM y su clave privada se cifran con una clave KMS administrada de AWS que tiene el alias aws/acm. Puede ver el ID de la clave con este alias en la consola de AWS KMS, en Claves administradas de AWS.

API Gateway no accede directamente a sus recursos de ACM. Utiliza AWS TLS Connection Manager para proteger las claves privadas de su certificado y acceder a ellas. Cuando usa su certificado de ACM para crear un nombre de dominio personalizado para API privadas de API Gateway, API Gateway asocia su certificado a AWS TLS Connection Manager. Para ello, se crea una concesión en AWS KMS para la clave administrada de AWS con el prefijo aws/acm. Una concesión es un instrumento de política que permite que TLS Connection Manager use claves KMS en operaciones criptográficas. La concesión le permite a la entidad principal beneficiaria (TLS Connection Manager) llamar a las operaciones de concesión especificadas en la clave KMS para descifrar la clave privada de su certificado. TLS Connection Manager utiliza entonces el certificado y la clave privada descifrada (texto sin formato) para establecer una conexión segura (sesión SSL/TLS) con los clientes de los servicios de API Gateway. Cuando el certificado se desvincula de un nombre de dominio personalizado de API Gateway para API privadas, la concesión se retira.

Si desea eliminar el acceso a la clave KMS, le recomendamos que sustituya o elimine el certificado del servicio usando la Consola de administración de AWS o el comando `update-service` en la AWS CLI.

Contexto de cifrado para API Gateway

Un contexto de cifrado es un conjunto de pares valor-clave opcional que contiene información contextual sobre los usos posibles de la clave privada. AWS KMS asocia el contexto de cifrado a los datos cifrados y lo utiliza como información autenticada adicional para permitir el cifrado autenticado.

Cuando las claves TLS se utilizan con API Gateway y TLS Connection Manager, el nombre del servicio de API Gateway se incluye en el contexto de cifrado utilizado para cifrar la clave en reposo. Para comprobar para qué nombre de dominio personalizado de API Gateway se utilizan su certificado y su clave privada, consulte el contexto de cifrado en sus registros de CloudTrail, tal como se muestra en la próxima sección, o bien la pestaña Recursos asociados de la consola de ACM.

Para descifrar los datos, se incluye el mismo contexto de cifrado en la solicitud. API Gateway utiliza el mismo contexto de cifrado en todas las operaciones criptográficas de AWS KMS, donde la clave es `aws:apigateway:arn` y el valor es el nombre de recurso de Amazon (ARN) del recurso `PrivateDomainName` de API Gateway.

El siguiente ejemplo muestra el contexto de cifrado en el resultado de una operación como `CreateGrant`.

```
"constraints": {
"encryptionContextEquals": {
"aws:acm:arn": "arn:aws:acm:us-west-2:859412291086:certificate/9177097a-
f0ae-4be1-93b1-19f911ea4f88",
"aws:apigateway:arn": "arn:aws:apigateway:us-west-2:859412291086:/domainnames/denytest-
part1.pdx.sahig.people.aws.dev+cbaeumzjhg"
}
},
"operations": [
"Decrypt"
],
"granteePrincipal": "tlsconnectionmanager.amazonaws.com"
```

Cifrado de datos en tránsito en Amazon API Gateway

Todas las API creadas con Amazon API Gateway solo exponen los puntos de conexión HTTPS. API Gateway no admite puntos de conexión no cifrados (HTTP).

API Gateway administra los certificados de los puntos de conexión de `execute-api` predeterminados. Si configura un nombre de dominio personalizado, especifique el certificado para el nombre de dominio. Como práctica recomendada, no fije certificados.

Para mayor seguridad, puede elegir una versión mínima del protocolo Transport Layer Security (TLS) que se va a aplicar a su dominio personalizado de API Gateway. Las API de WebSocket y las API HTTP solo admiten TLS 1.2. Para obtener más información, consulte Elección de una política de seguridad para el dominio personalizado en API Gateway.

También puede configurar una distribución de Amazon CloudFront con un certificado SSL personalizado en su cuenta y utilizarla con las API regionales. Luego puede configurar la política de seguridad para la distribución de CloudFront con TLS 1.1 o superior en función de los requisitos de seguridad y conformidad.

Para obtener más información sobre la protección de datos, consulte la entrada de blog relativa a Protección de sus API de REST en API Gateway y al modelo de responsabilidad compartida y GDPR de AWS en el blog de seguridad de AWS.

Privacidad del tráfico entre redes

Con Amazon API Gateway, puede crear API de REST privadas a las que solo se puede acceder desde su Amazon Virtual Private Cloud (VPC). La VPC utiliza un punto de enlace de la VPC de interfaz, que es una interfaz de red de punto de enlace que se crea en su VPC. Mediante las políticas de recursos, puede permitir o denegar el acceso a las API desde las VPC y los puntos de enlace de la VPC seleccionados, incluso entre diferentes cuentas de AWS. Cada punto de enlace se puede utilizar para tener acceso a varias API privadas. También puede utilizar Direct Connect para establecer una conexión entre una red en las instalaciones y Amazon VPC, y obtener acceso a la API privada a través de esa conexión. En cualquier caso, el tráfico dirigido a la API privada utilizará conexiones seguras y no saldrá de la red de Amazon, ya que está aislado de la red pública de Internet. Para obtener más información, consulte the section called “API de REST privadas”.

Identity and Access Management para Amazon API Gateway

AWS Identity and Access Management (IAM) es un Servicio de AWS que ayuda a los administradores a controlar de forma segura el acceso a los recursos de AWS. Los administradores de IAM controlan quién puede estar autenticado (iniciar sesión) y autorizado (tiene permisos) para utilizar recursos de API Gateway. IAM es un Servicio de AWS que se puede utilizar sin cargo adicional.

Temas

- Público
- Autenticación con identidades
- Administración del acceso con políticas
- Cómo funciona Amazon API Gateway con IAM
- Ejemplos de políticas basadas en identidades de Amazon API Gateway
- Ejemplos de políticas basadas en recursos de Amazon API Gateway
- Solución de problemas de identidad y acceso de Amazon API Gateway
- Uso de roles vinculados a servicios para API Gateway

Público

La forma de utilizar AWS Identity and Access Management (IAM) varía en función del rol:

- Usuario del servicio: solicite permisos al administrador si no puede acceder a las características (consulte Solución de problemas de identidad y acceso de Amazon API Gateway).
- Administrador del servicio: determine el acceso de los usuarios y envíe las solicitudes de permiso (consulte Cómo funciona Amazon API Gateway con IAM).
- Administrador de IAM: escribe las políticas para administrar el acceso (consulte Ejemplos de políticas basadas en identidades de Amazon API Gateway).

Autenticación con identidades

La autenticación es la manera de iniciar sesión en AWS mediante credenciales de identidad. Debe autenticarse como el Usuario raíz de la cuenta de AWS, como un usuario de IAM o asumiendo un rol de IAM.

Se puede iniciar sesión como una identidad federada con las credenciales de un origen de identidad, como AWS IAM Identity Center (IAM Identity Center), la autenticación de inicio de sesión único o las credenciales de Google/Facebook. Para obtener más información sobre el inicio de sesión, consulte Cómo iniciar sesión en la Cuenta de AWS en la Guía del usuario de AWS Sign-In.

Para el acceso mediante programación, AWS proporciona un SDK y una CLI para firmar criptográficamente las solicitudes. Para obtener más información, consulte AWS Signature Version 4 para solicitudes de API en la Guía del usuario de IAM.

Usuario raíz de la Cuenta de AWS

Cuando se crea una Cuenta de AWS, se comienza con una identidad de inicio de sesión denominada usuario raíz de la Cuenta de AWS que tiene acceso completo a todos los Servicios de AWS y recursos. Se recomiendaencarecidamente que no utilice el usuario raíz para las tareas diarias. Para ver la lista completa de las tareas que requieren credenciales de usuario raíz, consulte Tareas que requieren credenciales de usuario raíz en la Guía del usuario de IAM.

Usuarios y grupos de IAM

Un usuario de IAM es una identidad con permisos específicos para una sola persona o aplicación. Recomendamos el uso de credenciales temporales en lugar de usuarios de IAM con credenciales de larga duración. Para obtener más información, consulte Exigir que los usuarios humanos utilicen la federación con un proveedor de identidades para acceder a AWS con credenciales temporales en la Guía del usuario de IAM.

Un grupo de IAM especifica un conjunto de usuarios de IAM y facilita la administración de los permisos para grupos grandes de usuarios. Para obtener más información, consulte Casos de uso para usuarios de IAM en la Guía del usuario de IAM.

Roles de IAM

Un Rol de IAM es una identidad con permisos específicos que proporciona credenciales temporales. Se puede asumir un rol cambiando de un usuario a un rol de IAM (consola) o llamando a una operación de la API de la AWS CLI o AWS. Para obtener más información, consulte Métodos para asumir un rol en la Guía del usuario de IAM.

Los roles de IAM son útiles para el acceso de usuario federado, los permisos de usuario de IAM temporales, el acceso entre cuentas, el acceso entre servicios y las aplicaciones que se ejecutan en Amazon EC2. Para obtener más información, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Administración del acceso con políticas

Para controlar el acceso en AWS, se crean políticas y se adjuntan a identidades o recursos de AWS. Una política define los permisos cuando se asocia a una identidad o un recurso. AWS evalúa estas políticas cuando una entidad principal realiza una solicitud. La mayoría de las políticas se almacenan en AWS como documentos JSON. Para obtener más información sobre los documentos de políticas de JSON, consulte Información general de políticas de JSON en la Guía del usuario de IAM.

Mediante las políticas, los administradores especifican quién tiene acceso a qué, definiendo qué entidad principal puede realizar acciones sobre qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Un administrador de IAM crea políticas de IAM y las agrega a roles, que los usuarios pueden asumir posteriormente. Las políticas de IAM definen permisos independientemente del método que se utilice para realizar la operación.

Políticas basadas en identidades

Las políticas basadas en identidad son documentos de política de permisos JSON que asocia a una identidad (usuario, grupo o rol). Estas políticas controlan qué acciones pueden realizar las identidades, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en la identidad, consulte Definición de permisos de IAM personalizados con políticas administradas por el cliente en la Guía del usuario de IAM.

Las políticas basadas en identidad pueden ser políticas insertadas (incrustadas directamente en una sola identidad) o políticas administradas (políticas independientes asociadas a varias identidades). Para obtener información sobre cómo elegir entre políticas administradas e insertadas, consulte Selección entre políticas administradas y políticas insertadas en la Guía del usuario de IAM.

Políticas basadas en recursos

Las políticas basadas en recursos son documentos de políticas JSON que se asocian a un recurso. Los ejemplos incluyen las Políticas de confianza de roles de IAM y las Políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Debe especificar una entidad principal en una política basada en recursos.

Las políticas basadas en recursos son políticas insertadas que se encuentran en ese servicio. No se puede utilizar políticas de IAM administradas de AWS en una política basada en recursos.

Listas de control de acceso (ACL)

Las listas de control de acceso (ACL) controlan qué entidades principales (miembros de cuentas, usuarios o roles) tienen permisos para acceder a un recurso. Las ACL son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

Amazon S3, AWS WAF y Amazon VPC son ejemplos de servicios que admiten las ACL. Para obtener más información sobre las ACL, consulta Información general de Lista de control de acceso (ACL) en la Guía para desarrolladores de Amazon Simple Storage Service.

Otros tipos de políticas

AWS admite tipos de políticas adicionales que pueden establecer el máximo de permisos concedidos por los tipos de políticas más frecuentes:

- Límites de permisos: establecen los permisos máximos que una política basada en identidad puede conceder a una entidad de IAM. Para obtener más información, consulte Límites de permisos para las entidades de IAM en la Guía del usuario de IAM.
- Políticas de control de servicios (SCP): especifican los permisos máximos para una organización o unidad organizativa en AWS Organizations. Para obtener más información, consulte Políticas de control de servicios en la Guía del usuario de AWS Organizations.
- Políticas de control de recursos (RCP): definen los permisos máximos disponibles para los recursos de las cuentas. Para obtener más información, consulte Políticas de control de recursos (RCP) en la Guía del usuario de AWS Organizations.
- Políticas de sesión: políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal para un rol o un usuario federado. Para obtener más información, consulte Políticas de sesión en la Guía del usuario de IAM.

Varios tipos de políticas

Cuando se aplican varios tipos de políticas a una solicitud, los permisos resultantes son más complicados de entender. Para obtener información acerca de cómo AWS decide si se debe permitir o no una solicitud cuando hay implicados varios tipos de políticas, consulte Lógica de evaluación de políticas en la Guía del usuario de IAM.

Cómo funciona Amazon API Gateway con IAM

Antes de utilizar IAM para administrar el acceso a API Gateway, debe saber qué características de IAM están disponibles para usarlas con API Gateway. Para obtener una vista de alto nivel de cómo API Gateway y otros servicios de AWS funcionan con IAM, consulte Servicios de AWS que funcionan con IAM en la Guía del usuario de IAM.

Temas

- Políticas basadas en identidades de API Gateway
- Políticas basadas en recursos de API Gateway
- Autorización basada en etiquetas de API Gateway
- Roles de IAM de API Gateway

Políticas basadas en identidades de API Gateway

Con las políticas basadas en identidades de IAM, puede especificar las acciones y los recursos permitidos o denegados, así como las condiciones en las que se permiten o deniegan las acciones. API Gateway admite acciones, claves de condiciones y recursos específicos. Para obtener más información sobre las acciones, recursos y claves de condiciones específicos de API Gateway, consulte Acciones, recursos y claves de condición de la administración de Amazon API Gateway y Acciones, recursos y claves de condición de la administración de Amazon API Gateway V2. Para obtener información sobre todos los elementos que utiliza en una política JSON, consulte IAM JSON Policy Elements Reference en la Guía del usuario de IAM.

En el siguiente ejemplo se muestra una política basada en identidades que permite a un usuario crear o actualizar solo API REST privadas. Para obtener más ejemplos, consulte the section called "Ejemplos de políticas basadas en identidades".

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ScopeToPrivateApis",
      "Effect": "Allow",
      "Action": [
        "apigateway:PATCH",
        "apigateway:POST",
        "apigateway:PUT"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/restapis",
        "arn:aws:apigateway:us-east-1::/restapis/??????????"
      ],
      "Condition": {
        "ForAllValues:StringEqualsIfExists": {
          "apigateway:Request/EndpointType": "PRIVATE",
          "apigateway:Resource/EndpointType": "PRIVATE"
        }
      }
    },
    {
      "Sid": "AllowResourcePolicyUpdates",
```

```
      "Effect": "Allow",
      "Action": [
          "apigateway:UpdateRestApiPolicy"
      ],
      "Resource": [
          "arn:aws:apigateway:us-east-1::/restapis/*"
      ]
    }
  ]
}
```

Acciones

El elemento `Action` de una política JSON describe las acciones que puede utilizar para permitir o denegar el acceso en una política.

Las acciones de políticas de API Gateway incluyen el siguiente prefijo antes de la acción: `apigateway:`. Las instrucciones de la política deben incluir un elemento `Action` o un elemento `NotAction`. API Gateway define su propio conjunto de acciones que describen las tareas que se pueden realizar con este servicio.

La expresión `Action` de administración de la API tiene el formato `apigateway:`*`action`*, donde *`acción`* es una de las siguientes acciones de API Gateway: GET, POST, PUT, DELETE, PATCH (para actualizar los recursos) o *, que representa todas las acciones anteriores.

Algunos ejemplos de la expresión `Action` incluyen:

- **`apigateway:*`** para todas las acciones de API Gateway.
- **`apigateway:GET`** solo para la acción GET en API Gateway.

Para especificar varias acciones en una única instrucción, sepárelas con comas del siguiente modo:

```
"Action": [
      "apigateway:action1",
      "apigateway:action2"
```

Para obtener información acerca de los verbos HTTP que se deben utilizar para operaciones específicas de API Gateway, consulte Amazon API Gateway Version 1 API Reference (API REST) y Amazon API Gateway Version 2 API Reference (API HTTP y WebSocket).

Para obtener más información, consulte the section called “Ejemplos de políticas basadas en identidades”.

Recursos

Los administradores pueden utilizar las políticas JSON de AWS para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Resource` de la política JSON especifica el objeto u objetos a los que se aplica la acción. Como práctica recomendada, especifique un recurso utilizando el Nombre de recurso de Amazon (ARN). Para las acciones que no admiten permisos por recurso, utilice un carácter comodín (*) para indicar que la instrucción se aplica a todos los recursos.

```
"Resource": "*"
```

Los recursos de API Gateway tienen el siguiente formato de ARN:

```
arn:aws:apigateway:region::resource-path-specifier
```

Por ejemplo, para especificar una API REST con el identificador *api-id* y sus subrecursos, como los autorizadores en la instrucción, utilice el siguiente ARN:

```
"Resource": "arn:aws:apigateway:us-east-2::/restapis/api-id/*"
```

Para especificar todas las API REST y los subrecursos que pertenecen a una cuenta específica, utilice el carácter comodín (*):

```
"Resource": "arn:aws:apigateway:us-east-2::/restapis/*"
```

Para ver una lista de los tipos de recursos de API Gateway y sus ARN, consulte Referencia del nombre de recurso de Amazon (ARN) de API Gateway.

Claves de condición

Los administradores pueden utilizar las políticas JSON de AWS para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Condition` especifica cuándo se ejecutan las instrucciones en función de criterios definidos. Puede crear expresiones condicionales que utilizan operadores de condición, tales como

igual o menor que, para que la condición de la política coincida con los valores de la solicitud. Para ver todas las claves de condición globales de AWS, consulte Claves de contexto de condición globales de AWS en la Guía del usuario de IAM.

API Gateway define su propio conjunto de claves de condición y también admite el uso de algunas claves de condición globales. Para ver una lista de claves de condición de API Gateway, consulte Condition Keys for Amazon API Gateway en la Guía del usuario de IAM. Para obtener más información acerca de las acciones y los recursos con los que puede utilizar una clave de condición, consulte Actions Defined by Amazon API Gateway.

Para obtener información sobre el etiquetado, incluido el control de acceso basado en atributos, consulte Etiquetado.

Ejemplos

Para ver ejemplos de políticas basadas en identidades de API Gateway, consulte Ejemplos de políticas basadas en identidades de Amazon API Gateway.

Políticas basadas en recursos de API Gateway

Las políticas basadas en recursos son documentos de política JSON que especifican qué acciones puede realizar una entidad principal especificada en el recurso de API Gateway y bajo qué condiciones. API Gateway admite políticas de permisos basadas en recursos para las API de REST. Las políticas de recursos se utilizan para controlar quién puede invocar una API REST. Para obtener más información, consulte the section called “Uso de políticas de recursos de API Gateway”.

Ejemplos

Para ver ejemplos de políticas basadas en recursos de API Gateway, consulte Ejemplos de políticas de recursos de API Gateway.

Autorización basada en etiquetas de API Gateway

Puede asociar etiquetas a recursos de API Gateway o transmitir etiquetas en una solicitud a API Gateway. Para controlar el acceso en función de etiquetas, debe proporcionar información de las etiquetas en el elemento de condición de una política utilizando las claves de condición `apigateway:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* o `aws:TagKeys`. Para obtener más información acerca de las etiquetas para recursos de API Gateway, consulte the section called “Control de acceso basado en atributos”.

Para consultar un ejemplo de política basada en la identidad para limitar el acceso a un recurso en función de las etiquetas de ese recurso, consulte Uso de etiquetas para controlar el acceso a los recursos API de REST de API Gateway.

Roles de IAM de API Gateway

Un rol de IAM es una entidad de la cuenta de AWS que dispone de permisos específicos.

Uso de credenciales temporales con API Gateway

Puede utilizar credenciales temporales para iniciar sesión con federación, asumir un rol de IAM o asumir un rol de acceso entre cuentas. Las credenciales de seguridad temporales se obtienen mediante una llamada a operaciones de la API de AWS STS, como AssumeRole o GetFederationToken.

API Gateway admite el uso de credenciales temporales.

Roles vinculados a servicios

Los roles vinculados a servicios permiten a los servicios de AWS obtener acceso a los recursos de otros servicios para completar una acción en su nombre. Los roles vinculados a servicios aparecen en la cuenta de IAM y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.

API Gateway admite roles vinculados a servicios. Para obtener información sobre cómo crear o administrar roles vinculados a servicios de API Gateway, consulte Uso de roles vinculados a servicios para API Gateway.

Roles de servicio

Un servicio puede asumir un rol de servicio en su nombre. Un rol de servicio permite que el servicio obtenga acceso a los recursos de otros servicios para completar una acción en su nombre. Los roles de servicio aparecen en la cuenta de IAM y son propiedad de la cuenta, por lo que un administrador IAM puede cambiar los permisos para este rol. Sin embargo, hacerlo podría deteriorar la funcionalidad del servicio.

API Gateway admite roles de servicio.

Ejemplos de políticas basadas en identidades de Amazon API Gateway

De forma predeterminada, los usuarios y roles de IAM no tienen permiso para crear ni modificar recursos de API Gateway. Tampoco pueden realizar tareas mediante la Consola de administración

de AWS, la AWS CLI o los AWS SDK. Un administrador de IAM debe crear políticas de IAM que concedan permisos a los usuarios y a los roles para realizar operaciones de la API concretas en los recursos especificados que necesiten. El administrador debe adjuntar esas políticas a los usuarios o grupos de IAM que necesiten esos permisos.

Para obtener información sobre cómo crear políticas de IAM, consulte Creating Policies on the JSON Tab en la Guía del usuario de IAM. Para obtener información sobre las acciones, los recursos y las condiciones específicas de API Gateway, consulte Acciones, recursos y claves de condición de la administración de Amazon API Gateway y Acciones, recursos y claves de condición de la administración de Amazon API Gateway V2.

Temas

- Prácticas recomendadas sobre las políticas
- Cómo permitir a los usuarios consultar sus propios permisos
- Permisos sencillos de lectura
- Crear solo autorizadores REQUEST o JWT
- Requerir que el punto de enlace predeterminado execute-api esté deshabilitado
- Permitir a los usuarios crear o actualizar solo API REST privadas
- Requerir que las rutas API tengan autorización
- Evitar que un usuario cree o actualice un enlace VPC
- Ejemplos de políticas para utilizar reglas de enrutamiento

Prácticas recomendadas sobre las políticas

Las políticas basadas en identidades determinan si alguien puede crear, acceder o eliminar los recursos de API Gateway de la cuenta. Estas acciones pueden generar costos adicionales para su Cuenta de AWS. Siga estas directrices y recomendaciones al crear o editar políticas basadas en identidades:

- Comience a utilizar las políticas administradas de AWS y avance hacia permisos de privilegios mínimos. Para empezar a conceder permisos a los usuarios y cargas de trabajo, utilice las políticas administradas de AWS que otorgan permisos para muchos casos de uso comunes. Están disponibles en su Cuenta de AWS. Se recomienda definir políticas administradas por el cliente de AWS específicas para sus casos de uso a fin de reducir aún más los permisos. Con el fin de obtener más información, consulte las políticas administradas por AWS o las políticas administradas por AWS para funciones de tarea en la Guía de usuario de IAM.

- Aplique permisos de privilegio mínimo: cuando establezca permisos con políticas de IAM, conceda solo los permisos necesarios para realizar una tarea. Para ello, debe definir las acciones que se pueden llevar a cabo en determinados recursos en condiciones específicas, también conocidos como permisos de privilegios mínimos. Con el fin de obtener más información sobre el uso de IAM para aplicar permisos, consulte Políticas y permisos en IAM en la Guía del usuario de IAM.
- Utilice condiciones en las políticas de IAM para restringir aún más el acceso: puede agregar una condición a sus políticas para limitar el acceso a las acciones y los recursos. Por ejemplo, puede escribir una condición de políticas para especificar que todas las solicitudes deben enviarse utilizando SSL. También puede usar condiciones para conceder acceso a acciones de servicios si se emplean a través de un Servicio de AWS determinado como, por ejemplo, CloudFormation. Para obtener más información, consulte Elementos de la política de JSON de IAM: Condición en la Guía del usuario de IAM.
- Utiliza el analizador de acceso de IAM para validar las políticas de IAM con el fin de garantizar la seguridad y funcionalidad de los permisos: el analizador de acceso de IAM valida políticas nuevas y existentes para que respeten el lenguaje (JSON) de las políticas de IAM y las prácticas recomendadas de IAM. El analizador de acceso de IAM proporciona más de 100 verificaciones de políticas y recomendaciones procesables para ayudar a crear políticas seguras y funcionales. Para más información, consulte Validación de políticas con el Analizador de acceso de IAM en la Guía del usuario de IAM.
- Solicite la autenticación multifactor (MFA): si se encuentra en una situación en la que necesite usuarios raíz o de IAM en su Cuenta de AWS, active la MFA para obtener una mayor seguridad. Para exigir la MFA cuando se invoquen las operaciones de la API, añada condiciones de MFA a sus políticas. Para más información, consulte Acceso seguro a la API con MFA en la Guía del usuario de IAM.

Para obtener más información sobre las prácticas recomendadas de IAM, consulte Prácticas recomendadas de seguridad en IAM en la Guía del usuario de IAM.

Cómo permitir a los usuarios consultar sus propios permisos

En este ejemplo, se muestra cómo podría crear una política que permita a los usuarios de IAM ver las políticas administradas e insertadas que se asocian a la identidad de sus usuarios. Esta política incluye permisos para llevar a cabo esta acción en la consola o mediante programación con la AWS CLI o la API de AWS.

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Permisos sencillos de lectura

Esta política de ejemplo otorga a un usuario permiso para obtener información acerca de todos los recursos de una API HTTP o WebSocket con el identificador de `a123456789` en la región de AWS us-east-1. El recurso `arn:aws:apigateway:us-east-1::/apis/a123456789/*` incluye todos los subrecursos de la API, como autorizadores e implementaciones.

JSON

```
{
  "Version":"2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "apigateway:GET"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis/a123456789/*"
      ]
    }
  ]
}
```

Crear solo autorizadores REQUEST o JWT

Esta política de ejemplo permite a un usuario crear API con solo autorizadores `REQUEST` o `JWT`, incluso mediante import. En la sección `Resource` de la política, `arn:aws:apigateway:us-east-1::/apis/??????????` requiere que los recursos tengan un máximo de 10 caracteres, lo que excluye los subrecursos de una API. Este ejemplo utiliza `ForAllValues` en la sección `Condition` porque los usuarios pueden crear varios autorizadores a la vez mediante la importación de una API.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "OnlyAllowSomeAuthorizerTypes",
      "Effect": "Allow",
      "Action": [
        "apigateway:PUT",
        "apigateway:POST",
        "apigateway:PATCH"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis",
        "arn:aws:apigateway:us-east-1::/apis/??????????",
        "arn:aws:apigateway:us-east-1::/apis/*/authorizers",
        "arn:aws:apigateway:us-east-1::/apis/*/authorizers/*"
      ],
```

```
      "Condition": {
        "ForAllValues:StringEqualsIfExists": {
          "apigateway:Request/AuthorizerType": [
            "REQUEST",
            "JWT"
          ]
        }
      }
    }
  ]
}
```

Requerir que el punto de enlace predeterminado **execute-api** esté deshabilitado

Esta política de ejemplo permite a los usuarios crear, actualizar o importar una API, con el requisito de que `DisableExecuteApiEndpoint` es `true`. Cuando `DisableExecuteApiEndpoint` es `true`, los clientes no pueden utilizar el punto de enlace predeterminado `execute-api` para invocar una API.

Utilizamos la condición `BoolIfExists` para manejar una llamada para actualizar una API que no tiene la clave de condición `DisableExecuteApiEndpoint` completa. Cuando un usuario intenta crear o importar una API, la clave de condición `DisableExecuteApiEndpoint` siempre se completa.

Debido a que el recurso `apis/*` también captura subrecursos, como autorizadores o métodos, explícitamente lo aplicamos a las API con una instrucción `Deny`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "DisableExecuteApiEndpoint",
      "Effect": "Allow",
      "Action": [
        "apigateway:PATCH",
        "apigateway:POST",
        "apigateway:PUT"
      ],
```

```
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis",
        "arn:aws:apigateway:us-east-1::/apis/*"
      ],
      "Condition": {
        "BoolIfExists": {
          "apigateway:Request/DisableExecuteApiEndpoint": true
        }
      }
    },
    {
      "Sid": "ScopeDownToJustApis",
      "Effect": "Deny",
      "Action": [
        "apigateway:PATCH",
        "apigateway:POST",
        "apigateway:PUT"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis/*/*"
      ]
    }
  ]
}
```

Permitir a los usuarios crear o actualizar solo API REST privadas

En esta política de ejemplo se utiliza una clave de condición para requerir que un usuario cree solo API `PRIVATE` y para evitar actualizaciones que puedan cambiar una API de `PRIVATE` a otro tipo, como `REGIONAL`.

Utilizamos `ForAllValues` para requerir que cada `EndpointType` que se agregue a una API sea `PRIVATE`. Utilizamos una clave de condición de recurso para permitir cualquier actualización de una API siempre que sea `PRIVATE`. `ForAllValues` se aplica solo si una clave de condición está presente.

Utilizamos la herramienta de coincidencia no codiciosa (?) para hacer coincidir explícitamente los ID de API para evitar que se permitan recursos que no sean API, como los autorizadores.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ScopePutToPrivateApis",
            "Effect": "Allow",
            "Action": [
                "apigateway:PUT"
            ],
            "Resource": [
                "arn:aws:apigateway:us-east-1::/restapis",
                "arn:aws:apigateway:us-east-1::/restapis/??????????"
            ],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "apigateway:Resource/EndpointType": "PRIVATE"
                }
            }
        },
        {
            "Sid": "ScopeToPrivateApis",
            "Effect": "Allow",
            "Action": [
                "apigateway:DELETE",
                "apigateway:PATCH",
                "apigateway:POST"
            ],
            "Resource": [
                "arn:aws:apigateway:us-east-1::/restapis",
                "arn:aws:apigateway:us-east-1::/restapis/??????????"
            ],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "apigateway:Request/EndpointType": "PRIVATE",
                    "apigateway:Resource/EndpointType": "PRIVATE"
                }
            }
        },
        {
            "Sid": "AllowResourcePolicyUpdates",
            "Effect": "Allow",
```

```
            "Action": [
                "apigateway:UpdateRestApiPolicy"
            ],
            "Resource": [
                "arn:aws:apigateway:us-east-1::/restapis/*"
            ]
        }
    ]
}
```

Requerir que las rutas API tengan autorización

Esta política hace que los intentos de crear o actualizar una ruta (incluso mediante import) falle si la ruta no tiene autorización. `ForAnyValue` lo evalúa como falso si la clave no está presente, tal como cuando no se crea o actualiza una ruta. Utilizamos `ForAnyValue` porque se pueden crear varias rutas a través de la importación.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowUpdatesOnApisAndRoutes",
      "Effect": "Allow",
      "Action": [
        "apigateway:POST",
        "apigateway:PATCH",
        "apigateway:PUT"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis",
        "arn:aws:apigateway:us-east-1::/apis/??????????",
        "arn:aws:apigateway:us-east-1::/apis/*/routes",
        "arn:aws:apigateway:us-east-1::/apis/*/routes/*"
      ]
    },
    {
      "Sid": "DenyUnauthorizedRoutes",
      "Effect": "Deny",
      "Action": [
```

```
        "apigateway:POST",
        "apigateway:PATCH",
        "apigateway:PUT"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/apis",
        "arn:aws:apigateway:us-east-1::/apis/*"
      ],
      "Condition": {
        "ForAnyValue:StringEqualsIgnoreCase": {
          "apigateway:Request/RouteAuthorizationType": "NONE"
        }
      }
    }
  ]
}
```

Evitar que un usuario cree o actualice un enlace VPC

Esta política evita que un usuario cree o actualice un enlace VPC. Un enlace de VPC le permite exponer los recursos dentro de una Amazon VPC a clientes que están fuera de la VPC.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "DenyVPCLink",
      "Effect": "Deny",
      "Action": [
        "apigateway:POST",
        "apigateway:PUT",
        "apigateway:PATCH"
      ],
      "Resource": [
        "arn:aws:apigateway:us-east-1::/vpclinks",
        "arn:aws:apigateway:us-east-1::/vpclinks/*"
      ]
    }
```

```
  ]
}
```

Ejemplos de políticas para utilizar reglas de enrutamiento

En los siguientes ejemplos de políticas se muestra cómo utilizar las claves de condición de RoutingRule para controlar cómo los usuarios pueden enrutar el tráfico desde los nombres de dominio personalizados a las API de REST. Puede utilizar estos ejemplos para crear políticas detalladas sobre el tipo de reglas de enrutamiento que pueden crear los usuarios. Para obtener más información, consulte the section called “Reglas de enrutamiento”.

Cómo impedir que un usuario cambie la forma en que un nombre de dominio personalizado enruta una solicitud

Esta política impide que un usuario cree o actualice `BasePathMapping`, `ApiMapping` o `RoutingRule`. Todos estos recursos pueden cambiar la forma en que un nombre de dominio personalizado enruta las solicitudes a las API.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAccessBasePathMappingsApiMappingsRoutingRules",
      "Effect": "Deny",
      "Action": "apigateway:*",
      "Resource": [
        "arn:aws:apigateway:us-east-1::/domainnames/example.com/basepathmappings/
*",
        "arn:aws:apigateway:us-east-1::/domainnames/example.com/apimappings/*",
        "arn:aws:apigateway:us-east-1:111122223333:/domainnames/example.com/
routingrules/*"
      ]
    }
  ]
}
```

Autorización a un usuario para actualizar una regla de enrutamiento en determinadas prioridades

Esta política permite a un usuario actualizar solo una regla de enrutamiento a una prioridad entre 1001 y 2000. Puede utilizar esta regla para separar las reglas de producción de las reglas de menor prioridad y, a continuación, permitir a los usuarios modificar las reglas de menor prioridad sin afectar las reglas de producción.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "UpdatingRoutingRulePriorityBetween1001And2000",
      "Effect": "Allow",
      "Action": "apigateway:UpdateRoutingRule",
      "Resource": "arn:aws:apigateway:us-east-1:111122223333:/domainnames/
example.com/routingrules/*",
      "Condition": {
        "NumericGreaterThanEquals": {
          "apigateway:Resource/Priority": 1001,
          "apigateway:Request/Priority": 1001
        },
       "NumericLessThanEquals": {
          "apigateway:Resource/Priority": 2000,
          "apigateway:Request/Priority": 2000
        }
      }
    }
  ]
}
```

Autorización a un usuario para actualizar una regla de enrutamiento o la asignación de ruta base para un valor de ruta base determinado

Esta política permite a un usuario actualizar solo una asignación de ruta base para cualquier ruta base que comience por `orders` o actualizar una regla de enrutamiento que coincida con una ruta base que comience por `orders`. En esta política, un usuario puede actualizar una asignación de ruta base o una regla de enrutamiento para `orders/create` o `orders123`, pero no para `payment/orders`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
      {
        "Sid": "AllowUpdateRoutingRuleUnderPathOrders",
        "Effect": "Allow",
        "Action": "apigateway:UpdateRoutingRule",
        "Resource": "arn:aws:apigateway:us-east-1:111122223333:/domainnames/
example.com/routingrules/*",
        "Condition": {
            "ForAllValues:StringLike": {
                "apigateway:Request/ConditionBasePaths": ["orders*"],
                "apigateway:Resource/ConditionBasePaths": ["orders*"]
            },
            "Null":{
                 "apigateway:Request/ConditionBasePaths":"false",
                 "apigateway:Resource/ConditionBasePaths":"false"
          }
        }
      }
  ]
}
```

Autorización a un usuario para que actualice el modo de enrutamiento a valores específicos

Esta política permite a un usuario actualizar solo el modo de enrutamiento a `API_MAPPING_ONLY` y `ROUTING_RULE_THEN_API_MAPPING`. Para obtener más información sobre el modo de enrutamiento, consulte the section called “Establecimiento del modo de enrutamiento para el nombre de dominio personalizado”.

JSON

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

```
            "Action": ["apigateway:PATCH"],
            "Resource": "arn:aws:apigateway:us-east-1::/domainnames/example.com",
               "Condition": {
                "StringLike": {
                    "apigateway:Request/RoutingMode":"*API_MAPPING*"
                }
            }
        }
    ]
}
```

Ejemplos de políticas basadas en recursos de Amazon API Gateway

Para ver ejemplos de política basada en recursos, consult the section called “Ejemplos de políticas de recursos de API Gateway”.

Solución de problemas de identidad y acceso de Amazon API Gateway

Utilice la siguiente información para diagnosticar y solucionar los problemas comunes que puedan surgir cuando se trabaja con API Gateway e IAM.

Temas

- No tengo autorización para realizar una acción en API Gateway.
- No tengo autorización para realizar la operación iam:PassRole
- Quiero permitir que personas ajenas a mi cuenta de AWS accedan a mis recursos de API Gateway

No tengo autorización para realizar una acción en API Gateway.

Si recibe un error que indica que no tiene autorización para realizar una acción, las políticas se deben actualizar para permitirle realizar la acción.

En el siguiente ejemplo, el error se produce cuando el usuario de IAM `mateojackson` intenta utilizar la consola para consultar los detalles acerca de un recurso ficticio *`my-example-widget`*, pero no tiene los permisos ficticios `apigateway:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 apigateway:GetWidget on resource: my-example-widget because no identity-based policy
 allows the GetWidget action
```

En este caso, la política del usuario `mateojackson` debe actualizarse para permitir el acceso al recurso *`my-example-widget`* mediante la acción `apigateway:`*`GetWidget`*.

Si necesita ayuda, contacte con su administrador de AWS. El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

No tengo autorización para realizar la operación iam:PassRole

Si recibe un error que indica que no está autorizado para realizar la acción `iam:PassRole`, las políticas se deben actualizar para permitir transferir un rol a API Gateway.

Algunos Servicios de AWS le permiten transferir un rol existente a dicho servicio en lugar de crear un nuevo rol de servicio o uno vinculado al servicio. Para ello, debe tener permisos para transferir el rol al servicio.

En el siguiente ejemplo, el error se produce cuando un usuario de IAM denominado `marymajor` intenta utilizar la consola para realizar una acción en API Gateway. Sin embargo, la acción requiere que el servicio cuente con permisos que concede un rol de servicio. Mary no tiene permisos para transferir el rol al servicio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

En este caso, las políticas de Mary se deben actualizar para permitirle realizar la acción `iam:PassRole`.

Si necesita ayuda, póngase en contacto con su administrador de AWS. El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

Quiero permitir que personas ajenas a mi cuenta de AWS accedan a mis recursos de API Gateway

Puede crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Se puede especificar una persona de confianza para que asuma el rol. En el caso de los servicios que admitan las políticas basadas en recursos o las listas de control de acceso (ACL), puede utilizar dichas políticas para conceder a las personas acceso a sus recursos.

Para obtener más información, consulte lo siguiente:

- Para saber si API Gateway admite estas características, consulte Cómo funciona Amazon API Gateway con IAM.
- Para obtener información acerca de cómo proporcionar acceso a los recursos de las Cuentas de AWS de su propiedad, consulte Proporcionar acceso a un usuario de IAM a otra Cuenta de AWS de la que es propietario en la Guía del usuario de IAM.
- Para obtener información acerca de cómo proporcionar acceso a sus recursos a Cuentas de AWS de terceros, consulte Proporcionar acceso a Cuentas de AWS que son propiedad de terceros en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso mediante una federación de identidades, consulte Proporcionar acceso a usuarios autenticados externamente (identidad federada) en la Guía del usuario de IAM.
- Para conocer sobre la diferencia entre las políticas basadas en roles y en recursos para el acceso entre cuentas, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Uso de roles vinculados a servicios para API Gateway

Amazon API Gateway utiliza Roles vinculados a servicios de AWS Identity and Access Management (IAM). Un rol vinculado a un servicio es un tipo único de rol de IAM que está vinculado directamente a API Gateway. Los roles vinculados a servicios están predefinidos por API Gateway e incluyen todos los permisos que el servicio requiere para llamar a otros servicios de AWS en su nombre.

Un rol vinculado a un servicio simplifica la configuración de API Gateway porque ya no tendrá que agregar manualmente los permisos necesarios. API Gateway define los permisos de sus roles vinculados a servicios y, a menos que esté definido de otra manera, solo API Gateway puede asumir sus roles. Los permisos definidos incluyen las políticas de confianza y de permisos, y que la política de permisos no se puede asociar a ninguna otra entidad de IAM

Solo puede eliminar un rol vinculado a un servicio después de eliminar los recursos relacionados. De esta forma, se protegen los recursos de API Gateway, ya que se evita que se puedan eliminar accidentalmente permisos de acceso a los recursos.

Para obtener información sobre otros servicios que admiten roles vinculados a servicios, consulte Servicios de AWS que funcionan con IAM y busque los servicios que muestran Sí en la columna Rol vinculado a servicios. Seleccione una opción Sí con un enlace para ver la documentación acerca del rol vinculado al servicio en cuestión.

Permisos de roles vinculados a servicios para API Gateway

API Gateway utiliza el rol vinculado a un servicio denominado AWSServiceRoleForAPIGateway: permite a API Gateway acceder a Elastic Load Balancing, Amazon Data Firehose y otros recursos de servicios en su nombre.

El rol vinculado a un servicio AWSServiceRoleForAPIGateway confía en que los siguientes servicios asuman el rol

- `ops.apigateway.amazonaws.com`

La política de permisos de rol permite que API Gateway realice las siguientes acciones en los recursos especificados:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:AddListenerCertificates",
                "elasticloadbalancing:RemoveListenerCertificates",
                "elasticloadbalancing:ModifyListener",
                "elasticloadbalancing:DescribeListeners",
                "elasticloadbalancing:DescribeLoadBalancers",
                "xray:PutTraceSegments",
                "xray:PutTelemetryRecords",
                "xray:GetSamplingTargets",
                "xray:GetSamplingRules",
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries",
                "servicediscovery:DiscoverInstances"
            ],
            "Resource": [
                "*"
            ]
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "firehose:DescribeDeliveryStream",
                "firehose:PutRecord",
                "firehose:PutRecordBatch"
            ],
            "Resource": "arn:aws:firehose:*:*:deliverystream/amazon-apigateway-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "acm:DescribeCertificate",
                "acm:GetCertificate"
            ],
            "Resource": "arn:aws:acm:*:*:certificate/*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateNetworkInterfacePermission",
            "Resource": "arn:aws:ec2:*:*:network-interface/*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:*:*:network-interface/*",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": [
                        "Owner",
                        "VpcLinkId"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:DeleteNetworkInterface",
                "ec2:AssignPrivateIpAddresses",
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
```

```
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeNetworkInterfaceAttribute",
                "ec2:DescribeVpcs",
                "ec2:DescribeNetworkInterfacePermissions",
                "ec2:UnassignPrivateIpAddresses",
                "ec2:DescribeSubnets",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "servicediscovery:GetNamespace",
            "Resource": "arn:aws:servicediscovery:*:*:namespace/*"
        },
        {
            "Effect": "Allow",
            "Action": "servicediscovery:GetService",
            "Resource": "arn:aws:servicediscovery:*:*:service/*"
        }
    ]
}
```

Debe configurar permisos para permitir a una entidad de IAM (como un usuario, grupo o función) crear, editar o eliminar la descripción de una función vinculada a un servicio. Para obtener más información, consulte Permisos de roles vinculados a servicios en la Guía del usuario de IAM..

Creación de un rol vinculado a un servicio para API Gateway

No necesita crear manualmente un rol vinculado a un servicio. Cuando crea una API, un nombre de dominio personalizado o un enlace de VPC en la Consola de administración de AWS, la AWS CLI o la API AWS, API Gateway crea de nuevo el rol vinculado a un servicio.

Si elimina este rol vinculado a servicio y necesita crearlo de nuevo, puede utilizar el mismo proceso para volver a crear el rol en su cuenta. Cuando crea una API, un nombre de dominio personalizado o un enlace de VPC, API Gateway crea de nuevo el rol vinculado a un servicio.

Modificación de un rol vinculado a un servicio para API Gateway

API Gateway no permite editar el rol vinculado a un servicio AWSServiceRoleForAPIGateway. Después de crear un rol vinculado a un servicio, no puede cambiarle el nombre, ya que varias entidades pueden hacer referencia a él. Sin embargo, puede editar la descripción del rol mediante IAM. Para obtener más información, consulte Editar un rol vinculado a servicios en la Guía del usuario de IAM..

Eliminación de un rol vinculado a un servicio para API Gateway

Si ya no necesita utilizar una característica o servicio que requiere un rol vinculado a un servicio, le recomendamos que elimine dicho rol. De esta forma no tiene una entidad no utilizada que no se monitoree ni mantenga de forma activa. Sin embargo, debe limpiar los recursos del rol vinculado al servicio antes de eliminarlo manualmente.

> **Note**
>
> Si el servicio de API Gateway utiliza el rol cuando intenta eliminar los recursos, es posible que no se pueda borrar. En tal caso, espere unos minutos e intente de nuevo la operación.

Para eliminar recursos de API Gateway utilizados por AWSServiceRoleForAPIGateway

1. Abra la consola de Amazon API Gateway en https://console.aws.amazon.com/apigateway.
2. Vaya a la API, el nombre de dominio personalizado o el enlace de la VPC que utiliza el rol vinculado al servicio.
3. Use la consola de para eliminar el recurso.
4. Repita el procedimiento para eliminar todas las API, nombres de dominio personalizados o vínculos de la VPC que utilicen el rol vinculado al servicio.

Cómo eliminar manualmente el rol vinculado a servicios mediante IAM

Utilice la consola de IAM, la AWS CLI o la API de AWS para eliminar el rol vinculado a un servicio AWSServiceRoleForAPIGateway. Para obtener más información, consulte Eliminación de un rol vinculado a servicios en la Guía del usuario de IAM.

Regiones que admiten roles vinculados a servicios de API Gateway

API Gateway admite el uso de roles vinculados a servicios en todas las regiones en las que el servicio está disponible. Para obtener más información, consulte Puntos de enlace del servicio de AWS.

Actualizaciones de API Gateway de las políticas administradas de AWS

Es posible consultar los detalles sobre las actualizaciones de las políticas administradas de AWS para API Gateway debido a que este servicio comenzó a realizar el seguimiento de estos cambios. Para obtener alertas automáticas sobre cambios en esta página, suscríbase a la fuente RSS en la página de historial de documentos de la API Gateway.

Cambio	Descripción	Fecha
Se ha añadido soporte `acm:GetCertificate` a la política de `AWSServiceRoleForAPIGateway` .	La política de `AWSServiceRoleForAPIGateway` ahora incluye permiso para llamar a la acción a la API de ACM `GetCertificate` .	12 de julio de 2021
API Gateway comenzó el seguimiento de los cambios	API Gateway comenzó el seguimiento de los cambios para las Políticas administradas por AWS.	12 de julio de 2021

Registro y monitoreo en Amazon API Gateway

El monitoreo es una parte importante para mantener la fiabilidad, la disponibilidad y el rendimiento de API Gateway y sus soluciones de AWS. Debe recopilar datos de monitoreo de todas las partes de su solución de AWS para que pueda depurar un error de varios puntos de una forma más fácil en caso de producirse. AWS proporciona varias herramientas para monitorear sus recursos de API Gateway y responder a posibles incidentes.

Amazon CloudWatch Logs

Para ayudarle a depurar problemas relacionados con la ejecución de solicitudes o el acceso de clientes a la API, puede habilitar CloudWatch Logs para registrar las llamadas a la API. Para obtener más información, consulte the section called “CloudWatch Logs”.

Alarmas de Amazon CloudWatch

Las alarmas de Amazon CloudWatch le permiten ver una sola métrica durante el período de tiempo que especifique. Si la métrica supera un límite determinado, se envía una notificación a un tema de Amazon Simple Notification Service o a una política de AWS Auto Scaling. Las alarmas de CloudWatch no invocan acciones cuando una métrica se encuentra en un estado determinado. En su lugar, el estado debe haber cambiado y debe mantenerse durante el número de periodos especificado. Para obtener más información, consulte the section called “Métricas de CloudWatch”.

Registro de acceso a Firehose

Para ayudar a depurar los problemas relacionados con el acceso de los clientes a la API, puede habilitar Firehose para registrar las llamadas a la API. Para obtener más información, consulte the section called “Firehose”.

AWS CloudTrail

CloudTrail proporciona un registro de las medidas adoptadas por un usuario, un rol o un servicio de AWS en API Gateway. Mediante la información recopilada por CloudTrail, puede determinar la solicitud que se realizó a API Gateway, la dirección IP de origen desde la que se realizó, quién la realizó y cuándo, etc. Para obtener más información, consulte the section called “Trabajar con CloudTrail”.

AWS X-Ray

X-Ray es un servicio de AWS que recopila datos sobre las solicitudes que la aplicación atiende y los utiliza para crear un mapa de servicios que puede usar para identificar problemas con la aplicación y oportunidades de optimización. Para obtener más información, consulte the section called “Configuración de AWS X-Ray”.

AWS Config

AWS Config proporciona una vista detallada de la configuración de los recursos de AWS de su cuenta. Puede observar las relaciones entre los recursos, obtener un historial de los cambios de configuración y comprobar cómo cambian las relaciones y configuraciones con el paso del

tiempo. Puede utilizar AWS Config para definir reglas que evalúen la conformidad de los datos de estas configuraciones de recursos. Las reglas de AWS Config representan las opciones de configuración ideales para sus recursos de API Gateway. Si un recurso infringe una regla y está marcado como no conforme, AWS Config puede avisarle mediante un tema de Amazon Simple Notification Service (Amazon SNS). Para obtener información, consulte the section called "Trabajo con AWS Config".

Registro de llamadas a las API de Amazon API Gateway con AWS CloudTrail

Amazon API Gateway está integrado con AWS CloudTrail, un servicio que proporciona un registro de las acciones realizadas por un usuario, un rol o un Servicio de AWS. CloudTrail captura las llamadas a la API de REST de servicio de API Gateway como eventos. Las llamadas capturadas incluyen las llamadas desde la consola de API Gateway y las llamadas de código a las API de servicio de API Gateway. Mediante la información recopilada por CloudTrail, puede determinar la solicitud que se realizó a API Gateway, la dirección IP desde la que se realizó, cuándo se realizó y detalles adicionales.

> **Note**
>
> TestInvokeAuthorizer y TestInvokeMethod no se registran en CloudTrail.

Cada entrada de registro o evento contiene información sobre quién generó la solicitud. La información de identidad del usuario le ayuda a determinar lo siguiente:

- Si la solicitud se realizó con las credenciales del usuario raíz o del usuario.
- Si la solicitud se realizó en nombre de un usuario de IAM Identity Center.
- Si la solicitud se realizó con credenciales de seguridad temporales de un rol o fue un usuario federado.
- Si la solicitud la realizó otro Servicio de AWS.

CloudTrail está activado en la Cuenta de AWS cuando usted crea la cuenta y tiene acceso automático al Historial de eventos de CloudTrail. El Historial de eventos de CloudTrail proporciona un registro visible e inmutable, que se puede buscar y descargar, de los últimos 90 días de eventos de gestión registrados en una Región de AWS. Para obtener más información, consulte Trabajar con el

historial de eventos de CloudTrail en la Guía del usuario de AWS CloudTrail. No se cobran cargos de CloudTrail por ver el Historial de eventos.

Para mantener un registro permanente de los eventos en su Cuenta de AWS más allá de los 90 días, cree un registro de seguimiento o un almacén de datos de eventos de CloudTrail Lake.

Registros de seguimiento de CloudTrail

Un registro de seguimiento permite a CloudTrail enviar archivos de registro a un bucket de Amazon S3. Todos los registros de seguimiento que cree con la Consola de administración de AWS son multirregionales. Puede crear un registro de seguimiento de una sola región o multirregionales mediante la AWS CLI. Se recomienda crear un registro de seguimiento multirregional, ya que registra actividad en todas las Regiones de AWS de su cuenta. Si crea un registro de seguimiento de una sola región, solo podrá ver los eventos registrados en la Región de AWS del registro de seguimiento. Para obtener más información acerca de los registros de seguimiento, consulte Creación de un registro de seguimiento para su Cuenta de AWS y Creación de un registro de seguimiento para una organización en la Guía del usuario de AWS CloudTrail.

Puede crear un registro de seguimiento para enviar una copia de los eventos de administración en curso en su bucket de Amazon S3 sin costo alguno desde CloudTrail; sin embargo, hay cargos por almacenamiento en Amazon S3. Para obtener más información sobre los precios de CloudTrail, consulte Precios de AWS CloudTrail. Para obtener información acerca de los precios de Amazon S3, consulte Precios de Amazon S3.

Almacenes de datos de eventos de CloudTrail Lake

CloudTrail Lake le permite ejecutar consultas basadas en SQL sobre los eventos. CloudTrail Lake convierte los eventos existentes en formato JSON basado en filas al formato ORC de Apache. ORC es un formato de almacenamiento en columnas optimizado para una recuperación rápida de datos. Los eventos se agregan en almacenes de datos de eventos, que son recopilaciones inmutables de eventos en función de criterios que se seleccionan aplicando selectores de eventos avanzados. Los selectores que se aplican a un almacén de datos de eventos controlan los eventos que perduran y están disponibles para la consulta. Para obtener más información acerca de CloudTrail Lake, consulte Trabajar con AWS CloudTrail Lake en la Guía del usuario de AWS CloudTrail.

Los almacenes de datos de eventos de CloudTrail Lake y las consultas generan costos adicionales. Cuando crea un almacén de datos de eventos, debe elegir la opción de precios que desee utilizar para él. La opción de precios determina el costo de la incorporación y el

almacenamiento de los eventos, así como el período de retención predeterminado y máximo del almacén de datos de eventos. Para obtener más información sobre los precios de CloudTrail, consulte Precios de AWS CloudTrail.

Eventos de administración de API Gateway en CloudTrail

Los eventos de administración proporcionan información sobre las operaciones de administración que se realizan en los recursos de su Cuenta de AWS. Se denominan también operaciones del plano de control. CloudTrail registra los eventos de administración de forma predeterminada.

Amazon API Gateway registra todas las acciones de API Gateway como eventos de administración, excepto TestInvokeAuthorizer y TestInvokeMethod. Para obtener una lista de las acciones de Amazon API Gateway que API Gateway registra en CloudTrail, consulte la Referencia de la API de Amazon API Gateway.

Ejemplo de evento de API Gateway

Un evento representa una única solicitud de cualquier origen e incluye información sobre la operación de la API solicitada, la fecha y la hora de la operación o los parámetros de la solicitud, entre otras cosas. Los archivos de registro de CloudTrail no rastrean el orden en la pila de las llamadas a la API públicas, por lo que los eventos no aparecen en un orden específico.

En el ejemplo siguiente se muestra un evento de CloudTrail que demuestra la acción `GetResource` de API Gateway:

```
{
    Records: [
        {
            eventVersion: "1.03",
            userIdentity: {
                type: "Root",
                principalId: "AKIAI44QH8DHBEXAMPLE",
                arn: "arn:aws:iam::123456789012:root",
                accountId: "123456789012",
                accessKeyId: "AKIAIOSFODNN7EXAMPLE",
                sessionContext: {
                    attributes: {
                        mfaAuthenticated: "false",
                        creationDate: "2015-06-16T23:37:58Z"
                    }
```

```
                }
            },
            eventTime: "2015-06-17T00:47:28Z",
            eventSource: "apigateway.amazonaws.com",
            eventName: "GetResource",
            awsRegion: "us-east-1",
            sourceIPAddress: "203.0.113.11",
            userAgent: "example-user-agent-string",
            requestParameters: {
                restApiId: "3rbEXAMPLE",
                resourceId: "5tfEXAMPLE",
                template: false
            },
            responseElements: null,
            requestID: "6d9c4bfc-148a-11e5-81b6-7577cEXAMPLE",
            eventID: "4d293154-a15b-4c33-9e0a-ff5eeEXAMPLE",
            readOnly: true,
            eventType: "AwsApiCall",
            recipientAccountId: "123456789012"
        },
        ... additional entries ...
    ]
}
```

Para obtener información sobre el contenido de los registros de CloudTrail, consulte Contenido de los registros de CloudTrail en la Guía del usuario de AWS CloudTrail.

Monitoreo de la configuración de la API de API Gateway con AWS Config

Puede utilizar AWS Config para registrar cambios de configuración realizados en los recursos de la API de API Gateway y enviar notificaciones en función de los cambios de recursos. Mantener un historial de los cambios de configuración de los recursos de API Gateway es útil para los casos de uso relacionados con la solución de problemas operativos, auditoría y conformidad.

AWS Config puede realizar el seguimiento de cambios realizados en:

- Configuración de etapas de la API, como por ejemplo:
  - configuración del clúster de caché
  - configuración de restricción
  - configuración del registro de acceso
  - la implementación activa definida en la etapa

- Configuración de la API, como por ejemplo:
  - configuración de punto de enlace
  - versión
  - protocolo
  - etiquetas

Además, la característica Reglas de AWS Config le permite definir las reglas de configuración y detectar automáticamente, realizar el seguimiento y alertar cuando se infringen esas reglas. Mediante el seguimiento de los cambios realizados a las propiedades de la configuración de recursos, también puede crear reglas de AWS Config activadas por cambios para los recursos de API Gateway y probar las configuraciones de recursos en función de las prácticas recomendadas.

Puede habilitar AWS Config en su cuenta mediante la consola de AWS Config o la AWS CLI. Seleccione los tipos de recursos en los cuales quiere realizar el seguimiento de los cambios. Si anteriormente estableció la configuración para que AWS Config registrara todos los tipos de recursos, esos recursos de API Gateway se registran de forma automática en la cuenta. El soporte para Amazon API Gateway en AWS Config está disponible en todas las regiones públicas AWS y AWS GovCloud (US). Para ver la lista completa de las regiones admitidas, consulte Puntos de conexión y cuotas de Amazon API Gateway en la Referencia general de AWS.

Temas

- Tipos de recursos admitidos
- Configuración de AWS Config
- Configuración AWS Config para registrar recursos de API Gateway
- Visualización de los detalles de configuración de API Gateway en la consola de AWS Config
- Evaluación de recursos de API Gateway mediante las reglas de AWS Config

Tipos de recursos admitidos

Los siguientes tipos de recursos de API Gateway se integran con AWS Config y se documentan en Tipos de recursos de AWS Config admitidos por AWS y relaciones de recursos:

- `AWS::ApiGatewayV2::Api` (WebSocket y API HTTP)
- `AWS::ApiGateway::RestApi` (API REST)
- `AWS::ApiGatewayV2::Stage` (WebSocket y etapa de API HTTP)

- `AWS::ApiGateway::Stage` (Etapa de API de REST)

Para obtener más información sobre AWS Config, consulte la Guía para desarrolladores de AWS Config. Para obtener información acerca de los precios, consulte la página de información sobre precios de AWS Config.

> **Important**
>
> Si cambia cualquiera de las siguientes propiedades de API una vez que se implementa la API, es necesario volver a implementar la API para propagar los cambios. De lo contrario, verá los cambios de atributo en la consola de AWS Config, pero la configuración de la propiedad anterior seguirá estando en vigor; el comportamiento de tiempo de ejecución de la API permanecerá inalterado.
>
> - **`AWS::ApiGateway::RestApi`** – `binaryMediaTypes`, `minimumCompressionSize`, `apiKeySource`
> - **`AWS::ApiGatewayV2::Api`** – `apiKeySelectionExpression`

Configuración de AWS Config

Para configurar AWS Config inicialmente, consulte los temas siguientes en la Guía de desarrollador de AWS Config.

- Configuración de AWS Config con la consola
- Configuración de AWS Config con la AWS CLI

Configuración AWS Config para registrar recursos de API Gateway

De forma predeterminada, AWS Config registra los cambios de configuración de todos los tipos de recursos regionales admitidos que descubre en la región en la que se ejecuta el entorno. Puede personalizar AWS Config de forma que únicamente registre los cambios de determinados tipos de recursos o los cambios de los recursos globales.

Para obtener más información acerca de los recursos regionales y globales, así como el procedimiento de personalización de la configuración de AWS Config, consulte Selección de los recursos que debe registrar AWS Config.

Visualización de los detalles de configuración de API Gateway en la consola de AWS Config

Puede utilizar la consola de AWS Config para buscar recursos de API Gateway y obtener datos actuales e históricos sobre sus configuraciones. El siguiente procedimiento muestra cómo buscar información sobre una API de API Gateway.

Para buscar un recurso de API Gateway en la consola de AWS Config

1. Abra la consola de AWS Config.
2. Elija Resources (Recursos).
3. En la página Resource inventory, elija Resources.
4. Abra el menú Resource type (Tipo de recurso), vaya a APIGateway o APIGatewayV2 y elija uno o varios tipos de recursos de API Gateway.
5. Elija Look up (Buscar).
6. Elija un ID de recurso en la lista de recursos que muestra AWS Config. AWS Config muestra detalles de configuración y otra información sobre el recurso que ha seleccionado.
7. Para ver todos los detalles de la configuración registrada, elija View Details (Ver detalles).

Para conocer otras maneras de buscar un recurso y ver información en esta página, consulte Visualización de historial y configuraciones de recursos de AWS en la Guía del desarrollador de AWS Config.

Evaluación de recursos de API Gateway mediante las reglas de AWS Config

Puede crear reglas de AWS Config que representarán la configuración ideal para sus recursos de API Gateway. Puede utilizar reglas administradas de AWS Config o definir reglas personalizadas. AWS Config realiza un seguimiento constante de los cambios de configuración de los recursos para determinar si estos cambios infringen cualquiera de las condiciones de las reglas. La consola de AWS Config muestra el estado de conformidad de sus reglas y recursos.

Si un recurso infringe una regla y está marcado como no conforme, AWS Config puede avisarle mediante un tema de la Guía de desarrollador de Amazon Simple Notification Service (Amazon SNS). Para consumir mediante programación los datos de estas alertas de AWS Config, utilice una cola de Amazon Simple Queue Service (Amazon SQS) como punto de enlace de notificación para el tema Amazon SNS.

Para obtener más información acerca de cómo configurar y utilizar reglas, consulte Evaluación de recursos con reglas en la Guía para desarrolladores de AWS Config.

Validación de conformidad para Amazon API Gateway

Nuestra nueva experiencia de registro de AWS no está diseñada para cargas de trabajo reguladas. Si utiliza nuestra nueva experiencia de registro de AWS, pero desea utilizar AWS para cargas de trabajo reguladas, puede registrarse en AWS (avanzado) o activar características avanzadas para su entorno de AWS.

Para saber si un Servicio de AWS está incluido en el alcance de programas de cumplimiento específicos, consulte Servicios de AWS incluidos por programa de cumplimiento y elija el programa de cumplimiento de su interés. Para obtener información general, consulte Programas de conformidad de AWS.

Puede descargar los informes de auditoría de terceros utilizando AWS Artifact. Para obtener más información, consulte Descarga de informes en AWS Artifact.

Su responsabilidad de conformidad al utilizar Servicios de AWS se determina en función de la confidencialidad de los datos, los objetivos de conformidad de la empresa, así como de la legislación y los reglamentos aplicables. Para obtener más información sobre la responsabilidad de cumplimiento al usar Servicios de AWS, consulte la Documentación de seguridad de AWS.

Resiliencia en Amazon API Gateway

La infraestructura global de AWS se compone de regiones y zonas de disponibilidad de AWS. AWS Las regiones proporcionan varias zonas de disponibilidad físicamente independientes y aisladas que se encuentran conectadas mediante redes con un alto nivel de rendimiento y redundancia, además de baja latencia. Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre las zonas sin interrupciones. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de uno o varios centros de datos.

Como servicio regional completamente administrado, API Gateway funciona en varias zonas de disponibilidad en cada región y utiliza la redundancia de las zonas de disponibilidad para minimizar los errores de infraestructura como una categoría de riesgo de disponibilidad. API Gateway se ha diseñado para recuperarse automáticamente de un error en una zona de disponibilidad.

Para obtener más información sobre las regiones y zonas de disponibilidad de AWS, consulte Infraestructura global de AWS.

Para evitar que las API se vean abrumadas por demasiadas solicitudes, API Gateway limita las solicitudes a las API. En concreto, API Gateway establece un ratio de solicitudes en estado estable y una ráfaga de envíos de solicitudes para todas las API de su cuenta. Puede configurar la limitación controlada personalizada para sus API. Para obtener más información, consulte Limitación de las solicitudes a sus API de REST para mejorar el rendimiento en API Gateway.

Puede utilizar las comprobaciones de estado de Route 53 para controlar la conmutación por error de DNS de una API de API Gateway en una región principal a una API de API Gateway en una región secundaria. Para ver un ejemplo, consulte the section called “recuperación ante errores a nivel de DNS”.

Seguridad de la infraestructura en Amazon API Gateway

Como se trata de un servicio administrado, Amazon API Gateway está protegido por la seguridad de red global de AWS. Para obtener información sobre los servicios de seguridad de AWS y sobre cómo AWS protege la infraestructura, consulte Seguridad en la nube de AWS. Para diseñar su entorno de AWS siguiendo las prácticas recomendadas de seguridad de infraestructura, consulte Protección de la infraestructura en Portal de seguridad de AWS Well-Architected Framework.

Puede utilizar llamadas a la API publicadas de AWS para obtener acceso a API Gateway a través de la red. Los clientes deben admitir lo siguiente:

- Seguridad de la capa de transporte (TLS). Exigimos TLS 1.2 y recomendamos TLS 1.3.
- Conjuntos de cifrado con confidencialidad directa total (PFS) como DHE (Ephemeral Diffie-Hellman) o ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). La mayoría de los sistemas modernos como Java 7 y posteriores son compatibles con estos modos.

Puede llamar a estas operaciones de la API desde cualquier ubicación de red, pero API Gateway admite políticas de acceso basadas en recursos, que pueden incluir restricciones en función de la dirección IP de origen. También puede utilizar las políticas basadas en recursos para controlar el acceso desde puntos de enlace específicos de Amazon Virtual Private Cloud (Amazon VPC) o VPC específicas. Este proceso aísla con eficacia el acceso de red a un recurso de API Gateway determinado solo desde la VPC específica de la red de AWS.

Análisis de vulnerabilidades en Amazon API Gateway

La configuración y los controles de TI son una responsabilidad compartida entre AWS y usted, nuestro cliente. Para obtener más información, consulte el modelo de responsabilidad compartida de AWS.

Prácticas recomendadas sobre seguridad para Amazon API Gateway

API Gateway proporciona un número de características de seguridad que debe tener en cuenta a la hora de desarrollar e implementar sus propias políticas de seguridad. Las siguientes prácticas recomendadas son directrices generales y no suponen una solución de seguridad completa. Puesto que es posible que estas prácticas recomendadas no sean adecuadas o suficientes para el entorno, considérelas como consideraciones útiles en lugar de como normas.

Implementación del acceso a los privilegios mínimos

Utilice las políticas de IAM para implementar el acceso a los privilegios mínimos para crear, leer, actualizar o eliminar una API de API Gateway. Para obtener más información, consulte Identity and Access Management para Amazon API Gateway. API Gateway ofrece varias opciones para controlar el acceso a las API que cree. Para obtener más información, consulte Control y administración del acceso a las API de REST en API Gateway, Control y administración del acceso a las API de WebSocket en API Gateway y Control del acceso a API HTTP con autorizadores de JWT en API Gateway.

Implementar registro

Utilice CloudWatch Logs o Amazon Data Firehose para registrar solicitudes a las API. Para obtener más información, consulte Supervisión de las API de REST en API Gateway, Configuración del registro de las API de WebSocket en API Gateway y Configuración del registro de API HTTP en API Gateway.

Implementación de alarmas de Amazon CloudWatch

Las alarmas de Amazon CloudWatch le permiten ver una sola métrica durante el período de tiempo que especifique. Si la métrica supera un límite determinado, se envía una notificación a un tema de Amazon Simple Notification Service o a una política de AWS Auto Scaling. Las alarmas de CloudWatch no invocan acciones cuando una métrica se encuentra en un estado determinado. En su lugar, el estado debe haber cambiado y debe mantenerse durante el número

de periodos especificado. Para obtener más información, consulte the section called “Métricas de CloudWatch”.

Habilitar AWS CloudTrail

CloudTrail proporciona un registro de las medidas adoptadas por un usuario, un rol o un servicio de AWS en API Gateway. Mediante la información recopilada por CloudTrail, puede determinar la solicitud que se realizó a API Gateway, la dirección IP de origen desde la que se realizó, quién la realizó y cuándo, etc. Para obtener más información, consulte the section called “Trabajar con CloudTrail”.

Habilitar AWS Config

AWS Config proporciona una vista detallada de la configuración de los recursos de AWS de su cuenta. Puede observar las relaciones entre los recursos, obtener un historial de los cambios de configuración y comprobar cómo cambian las relaciones y configuraciones con el paso del tiempo. Puede utilizar AWS Config para definir reglas que evalúen la conformidad de los datos de estas configuraciones de recursos. Las reglas de AWS Config representan las opciones de configuración ideales para sus recursos de API Gateway. Si un recurso infringe una regla y está marcado como no conforme, AWS Config puede avisarle mediante un tema de Amazon Simple Notification Service (Amazon SNS). Para obtener información, consulte the section called “Trabajo con AWS Config”.

Uso de AWS Security Hub CSPM

Monitoree el uso de API Gateway en relación con las mejores prácticas de seguridad con AWS Security Hub CSPM. Security Hub CSPM utiliza controles de seguridad para evaluar las configuraciones de los recursos y los estándares de seguridad para ayudarlo a cumplir con varios marcos de conformidad. Para obtener más información sobre el uso de Security Hub CSPM para evaluar los recursos de API Gateway, consulte Controles de Amazon API Gateway en la Guía del usuario de AWS Security Hub.

Etiquetado de recursos de API Gateway

Una etiqueta es un elemento de metadatos que usted o AWS asigna a un recurso de AWS. Cada etiqueta tiene dos partes:

- Una clave de etiqueta (por ejemplo, `CostCenter`, `Environment` o `Project`). Las claves de etiqueta distinguen entre mayúsculas y minúsculas.
- Un campo opcional denominado valor de etiqueta (por ejemplo, `111122223333` o `Production`). Omitir el valor de etiqueta es lo mismo que usar una cadena vacía. Al igual que las claves de etiqueta, los valores de etiqueta distinguen entre mayúsculas y minúsculas.

Las etiquetas le ayudan a hacer lo siguiente:

- Controle el acceso a sus recursos basándose en las etiquetas que se les asignan. El acceso se controla especificando las claves y los valores de etiqueta en las condiciones de una política de AWS Identity and Access Management (IAM). Para obtener más información acerca del control de acceso basado en etiquetas, consulte Control del acceso mediante etiquetas en la Guía del usuario de IAM.
- Realizar un seguimiento de los costos de AWS. Estas etiquetas se activan en el panel de Administración de facturación y costos de AWS. AWS usa las etiquetas para clasificar los costos y enviar un informe mensual de asignación de costos. Para obtener más información, consulte Uso de etiquetas de asignación de costos en la Guía del usuario de AWS Billing.
- Identificar y organizar sus recursos de AWS. Muchos servicios de AWS admiten el etiquetado, por lo que puede asignar la misma etiqueta a los recursos de diferentes servicios para indicar que los recursos están relacionados. Por ejemplo, podría asignar la misma etiqueta a una etapa de API Gateway que se asigne a una regla de CloudWatch Events.

Para obtener sugerencias sobre el uso de etiquetas, consulte el documento sobre Estrategias de etiquetado de AWS.

En las siguientes secciones se ofrece más información acerca de las etiquetas de Amazon API Gateway.

Temas

- Recursos de API Gateway que se pueden etiquetar
- Uso de etiquetas para controlar el acceso a los recursos API de REST de API Gateway

Recursos de API Gateway que se pueden etiquetar

Las etiquetas se pueden configurar en los siguientes recursos de API HTTP o API de WebSocket en la API V2 de Amazon API Gateway:

- `Api`
- `DomainName`
- `Stage`
- `VpcLink`

Además, las etiquetas se pueden configurar en los siguientes recursos de API REST en la API V1 de Amazon API Gateway:

- `ApiKey`
- `ClientCertificate`
- `DomainName`
- `DomainNameAccessAssociation`
- `RestApi`
- `Stage`
- `UsagePlan`
- `VpcLink`

Las etiquetas no se pueden establecer directamente en otros recursos. Sin embargo, en la API V1 de Amazon API Gateway, los recursos secundarios heredan las etiquetas que se usan en los recursos principales. Por ejemplo:

- Si una etiqueta se establece en un recurso `RestApi`, se hereda en los siguientes recursos secundarios de esa `RestApi` para el control de acceso basado en atributos:
  - `Authorizer`
  - `Deployment`
  - `Documentation`
  - `GatewayResponse`
  - `Integration`

  - `Method`
  - `Model`
  - `Resource`
  - `ResourcePolicy`
  - `Setting`
  - `Stage`
- Si una etiqueta se establece en `DomainName`, se hereda en cualquiera de los recursos `BasePathMapping`, `ApiMapping` y `RoutingRule` por debajo de ella.
- Si una etiqueta se establece en `UsagePlan`, se hereda en cualquiera de los recursos `UsagePlanKey` por debajo de ella.

> **Note**
>
> La herencia de etiquetas solo es aplicable al control de acceso basado en atributos. Por ejemplo, no se pueden usar etiquetas heredadas para supervisar los costos en AWS Cost Explorer. Para usar etiquetas para la asignación de costos, le recomendamos que cree etiquetas en los recursos secundarios, como el recurso `Stage`.
> API Gateway no devuelve etiquetas heredadas cuando se llama a GetTags para un recurso.

Herencia de etiquetas en la API V1 de Amazon API Gateway

Antes solo se podían establecer etiquetas en etapas. Ahora que también puede establecerlas en otros recursos, `Stage` puede recibir una etiqueta de dos formas:

- La etiqueta se puede configurar directamente en `Stage`.
- La etapa puede heredar la etiqueta de su principa `RestApi`.

Si una etapa recibe una etiqueta de ambas maneras, tiene prioridad la etiqueta que se había establecido directamente en la etapa. Por ejemplo, suponga que una etapa hereda las siguientes etiquetas de su API REST principal:

```
{
 'foo': 'bar',
 'x':'y'
```

```
}
```

Suponga que también tiene las siguientes etiquetas establecidas en ella directamente:

```
{
 'foo': 'bar2',
 'hello': 'world'
}
```

El efecto neto sería que la etapa tuviera las etiquetas siguientes, con estos valores:

```
{
 'foo': 'bar2',
 'hello': 'world'
 'x':'y'
}
```

Restricciones de etiquetas y convenciones de uso

Las siguientes restricciones y convenciones de uso se aplican al uso de etiquetas con recursos de API Gateway:

- Cada recurso puede tener un máximo de 50 etiquetas.
- Para cada recurso, cada clave de etiqueta debe ser única y solo puede tener un valor.
- La longitud máxima de la clave de etiqueta es de 128 caracteres Unicode en UTF-8.
- La longitud máxima del valor de etiqueta es de 256 caracteres Unicode en UTF-8.
- Los caracteres permitidos para claves y valores son letras, números y espacios representables en UTF-8, además de los siguientes caracteres: . : + = @ _ / - (guion). Los recursos de Amazon EC2 admiten cualquier carácter.
- Las claves y los valores de las etiquetas distinguen entre mayúsculas y minúsculas. Como práctica recomendada, decida una estrategia de uso de mayúsculas y minúsculas en las etiquetas e implemente esa estrategia sistemáticamente en todos los tipos de recursos. Por ejemplo, decida si se va a utilizar `Costcenter`, `costcenter` o `CostCenter` y utilice la misma convención para todas las etiquetas. Procure no utilizar etiquetas similares con un tratamiento de mayúsculas y minúsculas incoherente.
- El prefijo `aws:` está prohibido para las etiquetas; está reservado para su uso por parte de AWS. Las claves y valores de etiquetas que tienen este prefijo no se pueden editar. Las etiquetas que tengan este prefijo no cuentan para el límite de etiquetas por recurso.

Uso de etiquetas para controlar el acceso a los recursos API de REST de API Gateway

Las condiciones de las políticas de AWS Identity and Access Management forman parte de la sintaxis que se utiliza para especificar los permisos de los recursos de API Gateway. Para obtener más información sobre la especificación de políticas de IAM, consulte the section called “Usar permisos de IAM”. En API Gateway, los recursos pueden tener etiquetas, y algunas acciones pueden incluir etiquetas. Al crear una política de IAM, puede utilizar las claves de condición de etiqueta para controlar:

- Qué usuarios pueden realizar acciones en un recurso de API Gateway, basándose en las etiquetas que ya tiene el recurso.
- Qué etiquetas se pueden pasar en la solicitud de una acción.
- Si se pueden utilizar claves de etiqueta específicas en una solicitud.

El uso del control de acceso basado en atributos puede permitir un control más preciso que el control a nivel de API, así como un control más dinámico que el control de acceso basado en recursos. Se pueden crear políticas de IAM que permitan o no una operación basada en las etiquetas que se proporcionan en la solicitud (etiquetas de solicitud) o en las etiquetas del recurso en el que se está operando (etiquetas de recurso). En general, las etiquetas de recursos corresponden a recursos que ya existen. Las etiquetas de solicitud corresponden al momento en que se crean nuevos recursos.

Para la sintaxis y semántica completas de las claves de condición de etiquetas, consulte Control del acceso mediante etiquetas en la Guía del usuario de IAM.

Los siguientes ejemplos muestran cómo especificar condiciones de etiquetas en las políticas para los usuarios de API Gateway.

Limitar acciones en función de etiquetas de recursos

La siguiente política de ejemplo concede permiso a los usuarios para realizar todas las acciones en todos los recursos, siempre y cuando esos recursos no tengan la etiqueta `Environment` con un valor de `prod`.

JSON

```
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "apigateway:*",
      "Resource": "*"
    },
    {
      "Effect": "Deny",
      "Action": [
        "apigateway:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Environment": "prod"
        }
      }
    }
  ]
}
```

Permitir acciones en función de las etiquetas de recursos

La siguiente política de ejemplo permite a los usuarios realizar todas las acciones en los recursos de API Gateway, siempre y cuando esos recursos tengan la etiqueta `Environment` con un valor de `Development`. La instrucción `Deny` impide que el usuario cambie el valor de la etiqueta `Environment`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ConditionallyAllow",
      "Effect": "Allow",
      "Action": [
        "apigateway:*"
      ],
      "Resource": [
```

```
        "arn:aws:apigateway:*::*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Environment": "Development"
        }
      }
    },
    {
      "Sid": "AllowTagging",
      "Effect": "Allow",
      "Action": [
        "apigateway:*"
      ],
      "Resource": [
        "arn:aws:apigateway:*::/tags/*"
      ]
    },
    {
      "Sid": "DenyChangingTag",
      "Effect": "Deny",
      "Action": [
        "apigateway:*"
      ],
      "Resource": [
        "arn:aws:apigateway:*::/tags/*"
      ],
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:TagKeys": "Environment"
        }
      }
    }
  ]
}
```

Denegar operaciones de etiquetado

La siguiente política de ejemplo permite a un usuario realizar todas las acciones de API Gateway, excepto cambiar las etiquetas.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "apigateway:*"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Deny",
            "Action": [
                "apigateway:*"
            ],
            "Resource": "arn:aws:apigateway:*::/tags*"
        }
    ]
}
```

Permitir operaciones de etiquetado

La siguiente política de ejemplo permite a un usuario obtener todos los recursos de API Gateway y cambiar las etiquetas de esos recursos. Para obtener las etiquetas de un recurso, el usuario debe tener permisos `GET` para ese recurso. Para actualizar las etiquetas de un recurso, el usuario debe tener permisos `PATCH` para ese recurso.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "apigateway:GET",
```

```
                "apigateway:PUT",
                "apigateway:POST",
                "apigateway:DELETE"
            ],
            "Resource": [
                "arn:aws:apigateway:*::/tags/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "apigateway:GET",
                "apigateway:PATCH"
            ],
            "Resource": [
                "arn:aws:apigateway:*::*"
            ]
        }
    ]
}
```

Referencias de API

Amazon API Gateway proporciona API para la creación y la implementación de sus propias API HTTP y de WebSocket. Además, las API de API Gateway están disponibles en los AWS SDK estándar.

Si utiliza un lenguaje para el que existe un AWS SDK, puede que prefiera usar el SDK en lugar de utilizar las API de REST de API Gateway directamente. Los SDK simplifican la autenticación, se integran fácilmente en su entorno de desarrollo y proporcionan acceso sencillo a los comandos de API Gateway.

Aquí puede encontrar la documentación de referencia de API de REST de los AWS SDK y API Gateway:

- Herramientas para Amazon Web Services
- Referencia de la API de REST de Amazon API Gateway
- Referencia de la API de Amazon API Gateway de WebSocket y de HTTP

Puntos de conexión del servicio de API Gateway

Un punto de conexión es una URL que sirve como punto de entrada para un servicio web de AWS. API Gateway admite los siguientes tipos de puntos de conexión:

- the section called “Puntos de conexión IPv4”
- Puntos de conexión de pila doble (compatibles con IPv4 e IPv6)
- Puntos de conexión de FIPS

Al realizar una solicitud, puede especificar el punto de conexión que se va a utilizar. Si no especifica un punto de conexión, se utilizará de forma predeterminada el punto de conexión IPv4. Para utilizar un tipo de punto de conexión diferente, debe especificarlo en la solicitud. Para ver ejemplos prácticos, consulte the section called “Especificación de puntos de conexión”. Para ver una tabla de los puntos de conexión disponibles, consulte Puntos de conexión de Amazon API Gateway.

Puntos de conexión IPv4

Los puntos de conexión IPv4 solo son compatibles con el tráfico IPv4. Los puntos de conexión IPv4 están disponibles para todas las regiones.

Si especifica el punto de conexión general `apigateway.amazonaws.com`, utilizamos el punto de conexión para `us-east-1`. Para utilizar una región diferente, especifique su punto de conexión asociado. Por ejemplo, si especifica `apigateway.us-east-2.amazonaws.com` como punto de conexión, dirigimos su solicitud al punto de conexión `us-east-2`.

Los nombres de puntos de conexión IPv4 utilizan la siguiente convención de nomenclatura:

- `apigateway.`*`region`*`.amazonaws.com`

Por ejemplo, el nombre del punto de conexión IPv4 para la región `eu-west-1` es `apigateway.eu-west-1.amazonaws.com`.

Puntos de conexión de pila doble (IPv4 e IPv6)

Los puntos de conexión de pila doble admiten tráfico IPv4 e IPv6. Cuando realiza una solicitud a un punto de conexión de pila doble, la URL del punto de conexión se resuelve en una dirección IPv6 o IPv4, en función del protocolo que utilicen la red y el cliente.

Los nombres de puntos de conexión de doble pila utilizan la siguiente convención de nomenclatura:

- `apigateway.`*`region`*`.api.aws`

Por ejemplo, el nombre del punto de conexión de doble pila para la región `eu-west-1` es `apigateway.eu-west-1.api.aws`.

Especificación de puntos de conexión

En los siguientes ejemplos, se muestra cómo especificar un punto de conexión para la región `us-east-2` mediante la AWS CLI para la `apigateway`.

- Pila doble

```
aws apigateway get-rest-apis --region us-east-2 --endpoint-url https://apigateway.us-east-2.api.aws
```

- IPv4

```
aws apigateway get-rest-apis --region us-east-2 --endpoint-url https://apigateway.us-east-2.amazonaws.com
```

En los siguientes ejemplos, se muestra cómo especificar un punto de conexión para la región `us-east-2` mediante la AWS CLI para la `apigatewayv2`.

- Pila doble

```
aws apigatewayv2 get-apis --region us-east-2 --endpoint-url https://apigateway.us-east-2.api.aws
```

- IPv4

```
aws apigatewayv2 get-apis --region us-east-2 --endpoint-url https://apigateway.us-east-2.amazonaws.com
```

Uso de direcciones IPv6 en políticas de IAM

Si utiliza políticas de usuario de IAM o políticas de recursos de API Gateway para controlar el acceso a API Gateway o a cualquier API de API Gateway, confirme que las políticas estén actualizadas para incluir los rangos de direcciones IPv6. Las políticas que no se actualizan para gestionar las direcciones IPv6 pueden afectar al acceso del cliente a API Gateway cuando comiencen a utilizar el punto de conexión de pila doble. Para obtener más información acerca de cómo administrar los permisos de acceso con IAM, consulte Identity and Access Management para Amazon API Gateway

Las políticas de IAM que filtran direcciones IP utilizan operadores de condición de dirección IP. La siguiente política de identidad permite a las direcciones IP del rango `54.240.143.*` obtener información acerca de todos los recursos de una API HTTP o de WebSocket con el identificador de `a123456789`. A cualquier dirección IP fuera de este rango se le denegará el acceso a la información de todos los recursos de la API. Dado que todas las direcciones IPv6 están fuera del rango permitido, esta política evita que las direcciones IPv6 puedan acceder a información sobre la API.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "IPAllow",
      "Effect": "Allow",
      "Principal": "*",
```

```
      "Action": "apigateway:GET",
      "Resource": "arn:aws:apigateway:us-east-1::/apis/a123456789/*",
      "Condition": {
         "IpAddress": {"aws:SourceIp": "54.240.143.0/24"}
      }
    }
  ]
}
```

Puede modificar el elemento `Condition` de la política de la API para permitir los rangos de direcciones IPv4 (`54.240.143.0/24`) e IPv6 (`2001:DB8:1234:5678::/64`) como se muestra en el siguiente ejemplo. Puede utilizar el mismo tipo de bloque `Condition` que se muestra en el ejemplo para actualizar las políticas de usuario de IAM y políticas de recursos de API Gateway.

```
      "Condition": {
        "IpAddress": {
           "aws:SourceIp": [
             "54.240.143.0/24",
              "2001:DB8:1234:5678::/64"
            ]
        }
       }
```

Cuotas de Amazon API Gateway

Las siguientes cuotas se aplican a todos los tipos de API de Amazon API Gateway.

Cuotas de nivel de cuenta de API Gateway, por región

Las siguientes cuotas se aplican por cuenta, por región en Amazon API Gateway.

Recurso u operación	Cuota predeterminada	Se puede aumentar
Cuota de limitación controlada por cuenta y región para todas las API REST, las API de WebSocket y las API de devolución de llamada de WebSocket	10 000 solicitudes por segundo (RPS) con una capacidad de ráfaga adicional proporcionada por el algoritmo de bucket de tokens, utilizando una capacidad máxima de bucket de 5,000 solicitudes. * **Note** La cuota de ráfaga la determina el equipo del servicio de API Gateway en función de las cuotas de RPS globales de la cuenta en la región. No es una cuota que los clientes puedan controlar o para el que puedan solicitar cambios.	Sí
Límite de la cuota sin control de acceso por cuenta y región para un portal	250,000 solicitudes por segundo	No
Límite de la cuota con control de acceso por cuenta y región para un portal	10,000 solicitudes por segundo	No

*Para las siguientes regiones, la cuota de limitación predeterminada es de 2500 RPS y la cuota de ráfaga predeterminada es de 1250 RPS: África (Ciudad del Cabo), Europa (Milán), Asia-Pacífico (Yakarta), Medio Oriente (EAU), Asia-Pacífico (Hyderabad), Asia-Pacífico (Melbourne), Europa (España), Europa (Zúrich), Israel (Tel Aviv), Oeste de Canadá (Calgary) y Asia-Pacífico (Malasia), Asia-Pacífico (Tailandia) y México (centro).

Cuotas de API Gateway para crear, implementar y administrar una API

Las siguientes cuotas fijas se aplican a la creación, la implementación y la administración de una API en API Gateway, mediante la AWS CLI, la consola de API Gateway o la API REST de API Gateway y sus SDK. Estas cuotas no se pueden aumentar.

Acción	Cuota predeterminada	Se puede aumentar
CreateApiKey	Cinco solicitudes por segundo y por cuenta	No
CreateDeployment	Una solicitud cada cinco segundos por cuenta	No
CreateDocumentationVersion	Una solicitud cada 20 segundos por cuenta	No
CreateDomainName	Una solicitud cada 30 segundos por cuenta	No
CreateResource	Cinco solicitudes por segundo y por cuenta	No
CreateRestApi para API regional o privada	Una solicitud cada tres segundos por cuenta	No
CreateRestApi para API optimizada para el borde	Una solicitud cada 30 segundos por cuenta	No
CreateVpcLink (V2)	1 solicitud cada 15 segundos por cuenta	No

Acción	Cuota predeterminada	Se puede aumentar
DeleteApiKey	Cinco solicitudes por segundo y por cuenta	No
DeleteDomainName	Una solicitud cada 30 segundos por cuenta	No
DeleteResource	Cinco solicitudes por segundo y por cuenta	No
DeleteRestApi	Una solicitud cada 30 segundos por cuenta	No
GetResources	Cinco solicitudes cada dos segundos por cuenta	No
DeleteVpcLink (V2)	Una solicitud cada 30 segundos por cuenta	No
ImportDocumentationParts	Una solicitud cada 30 segundos por cuenta	No
ImportRestApi para API regional o privada	Una solicitud cada tres segundos por cuenta	No
ImportRestApi para API optimizada para el borde	Una solicitud cada 30 segundos por cuenta	No
PutRestApi	Una solicitud por segundo por cuenta	No
UpdateAccount	Una solicitud cada 20 segundos por cuenta	No
UpdateDomainName	Una solicitud cada 30 segundos por cuenta	No
UpdateUsagePlan	Una solicitud cada 20 segundos por cuenta	No

Acción	Cuota predeterminada	Se puede aumentar
Creación de un portal	Una solicitud cada tres segundos	No
Portal de actualizaciones	Dos solicitudes por minuto	No
Obtención de portal	10 solicitudes por segundo	No
Lista de portales	10 solicitudes por segundo	No
Portal de publicaciones	Dos solicitudes por minuto	No
DeletePortal	Dos solicitudes por minuto	No
PreviewPortal	Una solicitud cada tres segundos	No
DisablePortal	Dos solicitudes por minuto	No
GetPortalProduct	10 solicitudes por segundo	No
ListPortalProduct	5 solicitudes por segundo	No
CreatePortalProduct	2 solicitudes por segundo	No
UpdatePortalProduct	0.5 solicitudes por segundo	No
DeletePortalProduct	Una solicitud por segundo	No
PutPortalProductSharingPolicy	Una solicitud cada tres segundos	No
GetPortalProductSharingPolicy	10 solicitudes por segundo	No
DeletePortalProductSharingPolicy	Una solicitud cada tres segundos	No
CreateProductRestEndpointPage	Una solicitud cada tres segundos	No

Acción	Cuota predeterminada	Se puede aumentar
GetProductRestEndpointPage	10 solicitudes por segundo	No
UpdateProductRestEndpointPage	Una solicitud cada tres segundos	No
DeleteProductRestEndpointPage	Una solicitud cada tres segundos	No
ListProductRestEndpointPages	10 solicitudes por segundo	No
CreateProductPage	Una solicitud cada tres segundos	No
GetProductPage	10 solicitudes por segundo	No
UpdateProductPage	Una solicitud cada tres segundos	No
DeleteProductPage	Una solicitud cada tres segundos	No
ListProductPages	10 solicitudes por segundo	No
Otras operaciones	No hay cuota hasta la cuota total de la cuenta.	No
Total de operaciones	10 solicitudes por segundo con una cuota de ráfaga de 40 solicitudes por segundo.	No

Cuotas para configurar y ejecutar una API de REST en API Gateway

Las siguientes cuotas se aplican a la configuración y la ejecución de una API REST en Amazon API Gateway. Si la cuota es por API, solo se puede aumentar en API específicas, no en todas las API de una cuenta.

Para obtener información sobre las cuotas en el nivel de cuenta, consulte Cuotas y notas importantes.

Recurso u operación	Cuota predeterminada	Se puede aumentar
Nombres de dominio personalizados públicos por cuenta y región	120	Sí
Nombres de dominio personalizados privados por cuenta y región	50	Sí
API regionales por región	600	No
API optimizadas para bordes por región	120	No
API privadas por cuenta por región	600	No
Asociaciones de acceso a nombres de dominio por cuenta	100	Sí

Recurso u operación	Cuota predeterminada	Se puede aumentar
Mapeos de la API de varios niveles por dominio	200	No
Reglas de enrutamiento por dominio	50	Sí
Longitud, en caracteres, de la URL para una API optimizada para límites	8192	No
Longitud, en caracteres, de la URL para una API regional	10240	No
Longitud, en caracteres, de la URL para una API privada	8192	No
Longitud, en caracteres, de la política de recursos de API Gateway	8192	Sí
Claves de API por cuenta y por región	10000	No

Recurso u operación	Cuota predeterminada	Se puede aumentar
Certificados de cliente generados por API Gateway por cuenta y por región	60	Sí
Certificados de cliente de ACM por cuenta y por región	500	No
Autorizadores por API (AWS Lambda y Amazon Cognito)	10	Sí Para aumentar esta cuota, póngase en contacto con AWS Support Center

Recurso u operación	Cuota predeterminada	Se puede aumentar
Partes de documentación por API	2000	Sí Para aumentar esta cuota, póngase en contacto con AWS Support Center
Recursos por API	300	Sí
Etapas por API	10	Sí
Variables de etapa por etapa	100	No
Longitud, en caracteres, de la clave en una variable de etapa	64	No
Longitud, en caracteres, del valor en una variable de etapa	512	No
Planes de uso por cuenta y por región	300	Sí

Recurso u operación	Cuota predeterminada	Se puede aumentar
Planes de uso por clave de API	10	Sí
Enlaces de VPC por cuenta y por región	20	Sí
TTL de almacenamiento en caché de la API	300 segundos de forma predeterminada y configurable entre 0 y 3600 por el propietario de la API.	No para el límite superior (3600)
Tamaño de respuesta en caché	1048576 bytes. El cifrado de datos de la caché puede aumentar el tamaño del elemento que se almacena en la caché.	No
Tiempo de espera de integración para las API regionales	50 milisegundos - 29 segundos para todos los tipos de integraciones, incluidas las integraciones de Lambda, proxy de Lambda, HTTP, proxy de HTTP y AWS.	Sí *
Tiempo de espera de integración para API optimizadas para periferia	50 milisegundos - 29 segundos para todos los tipos de integraciones, incluidas las integraciones de Lambda, proxy de Lambda, HTTP, proxy de HTTP y AWS.	No
Tiempo de espera de integración para API privadas	50 milisegundos - 29 segundos para todos los tipos de integraciones, incluidas las integraciones de Lambda, proxy de Lambda, HTTP, proxy de HTTP y AWS.	Sí *

Recurso u operación	Cuota predeterminada	Se puede aumentar
Tamaño total combinado de todos los valores del encabezado, incluidos los nombres de los encabezados, los valores, los terminadores de línea y los espacios en blanco	20480 bytes	No
Tamaño total combinado de todos los valores del encabezado para una API privada	8000 bytes	No
Tamaño de carga	10 MB	No
Etiquetas por etapa	50	No
Número de iteraciones en un bucle `#foreach ... #end` en las plantillas de asignación	1 000	No
Tamaño máximo de plantilla de asignación	300 KB	No

Recurso u operación	Cuota predeterminada	Se puede aumentar
Tamaño de la plantilla de registro de acceso	3 KB	No
Longitud del ARN del método	1600 bytes. Si el método contiene un parámetro de ruta y un cliente usa un valor que supera la longitud del ARN, la API devolverá una respuesta `414 Request URI too long`.	No
Configuración de la limitación controlada a nivel del método para una etapa de un plan de uso	20	Sí Para aumentar esta cuota, póngase en contacto con AWS Support Center
Tamaño de modelo por API	400 KB	No
Número de certificados en un almacén de confianza	1000 certificados con un tamaño de objeto total de hasta 1 MB.	No
Tiempo de inactividad de conexión	310 segundos	No

Recurso u operación	Cuota predeterminada	Se puede aumentar
Tamaño máximo del archivo de la definición de la API cuando se usa restapi:import o restapi:put	6 MB	No

*No puede establecer el tiempo de espera de la integración en menos de 50 milisegundos. Puede aumentar el tiempo de espera de integración a más de 29 segundos, pero esto podría requerir una reducción de la cuota de limitación de nivel regional de la cuenta.

Prácticas recomendadas para reducir las cuotas

Las siguientes prácticas recomendadas pueden ayudar a reducir la cantidad actual de recursos para evitar aumentar la cuota. Asegúrese de que estas sugerencias funcionen para la arquitectura de la API.

API por región

Para reducir el número de API por región, exporte las API no utilizadas y, a continuación, elimínelas de API Gateway. Para obtener más información, consulte the section called “Exportación de una API de REST”.

Etapas por API

Para reducir el número de etapas por API, divida la API en varias API.

Recursos por API

Use rutas de `{proxy+}` para reducir la cantidad de recursos. Para obtener más información, consulte the section called “Configuración de integraciones de proxy con un recurso de proxy”.

Asignación de API

Para reducir la cantidad de asignaciones de API para un nombre de dominio personalizado, utilice asignaciones de API con niveles únicos, como `/prod`. Las asignaciones de la API con un solo nivel no cuentan para la cuota de asignaciones de la API.

Autorizadores por cada API

Para reducir el número de autorizadores por API, reutilícelos en todos los métodos de la API.

Partes de documentación por API

Para reducir el número de partes de documentación por API, utilice `ignore=documentation` al importar la API. Para obtener más información, consulte the section called “Importar la documentación de API”.

También puede usar la herencia de contenido para permitir que algunas partes de la documentación hereden contenido de una entidad de API de una especificación más general. Para obtener más información, consulte the section called “Representación de la documentación de la API en API Gateway”.

Longitud, en caracteres, de la política de recursos de API Gateway

Para reducir la longitud de una política de recursos, use AWS WAFV2 para proteger la API. Para obtener más información, consulte the section called “AWS WAF”.

Si la política contiene direcciones IP, también puede usar rangos en lugar de valores específicos para definir las direcciones IP.

Planes de uso por clave de API

Para reducir el número de planes de uso por clave de la API, utilice una clave de la API por plan de uso y asocie el plan de uso a varias API. No recomendamos compartir una clave de API entre varios planes de uso.

Cuotas para configurar y ejecutar una API HTTP en API Gateway

Las cuotas siguientes se aplican a la configuración y ejecución de una API HTTP en Amazon API Gateway. Si la cuota es por API, solo se puede aumentar en API específicas, no en todas las API de una cuenta. Para obtener información sobre las cuotas en el nivel de cuenta, consulte Cuotas y notas importantes

Recurso u operación	Cuota predeterminada	Se puede aumentar
Rutas por API	300	Sí

Recurso u operación	Cuota predeterminada	Se puede aumentar
Integraciones por API	300	No
Tiempo de espera de integración máximo	30 segundos	No
Etapas por API	10	Sí
Mapeos de la API de varios niveles por dominio	200 asignaciones de la API. Este límite no incluye la asignación de la API con niveles únicos, como `/prod`.	No
Etiquetas por etapa	50	No
Tamaño total combinado de la línea de la solicitud y los valores del encabezado	10 240 bytes	No
Tamaño de carga	10 MB	No
Dominios personalizados por cuenta y región	120	Sí
Tamaño de la plantilla de registro de acceso	3 KB	No
Entrada de registro de Amazon CloudWatch Logs	1 MB	No

Recurso u operación	Cuota predeterminada	Se puede aumentar
Autorizadores por cada API	10	Sí Para aumentar esta cuota, póngase en contacto con AWS Support Center
Destinatarios por autorizador	50	No
Ámbitos por ruta	10	No
Tiempo de espera para el punto de conexión de JSON Web Key Set	1500 ms	No
Tamaño de respuesta desde el punto de conexión de JSON Web Key Set	150 000 bytes	No

Recurso u operación	Cuota predeterminada	Se puede aumentar
Tiempo de espera para el punto de conexión de detección de OpenID Connect	1500 ms	No
Se agotó el tiempo de respuesta del autorizador de Lambda	10 000 ms	No
Enlaces de VPC por cuenta y por región	10	Sí
Subredes por enlace de VPC	10	Sí
Variables de etapa por etapa	100	No
Longitud, en caracteres, de la clave en una variable de etapa	64	No
Longitud, en caracteres, del valor en una variable de etapa	512	No

Prácticas recomendadas para reducir las cuotas

Las siguientes prácticas recomendadas pueden ayudar a reducir la cantidad actual de recursos para evitar aumentar la cuota. Asegúrese de que estas sugerencias funcionen para la arquitectura de la API.

API por región

Para reducir el número de API por región, exporte las API no utilizadas y, a continuación, elimínelas de API Gateway. Para obtener más información, consulte the section called “Exportación”.

Etapas por API

Para reducir el número de etapas por API, divida su API en varias API.

Autorizadores por cada API

Para reducir el número de autorizadores por API, reutilice los autorizadores en todos los métodos de la API.

Cuotas para configurar portales en API Gateway

Las siguientes cuotas se aplican a la creación de portales en API Gateway. Para obtener más información, consulte Portales de API Gateway.

Recurso u operación	Cuota predeterminada	Se puede aumentar
Portales por cuenta	15	Sí
Productos de portal por portal	200	No
Productos de portal por cuenta	500	No
Páginas de punto de conexión REST	40	Sí

Recurso u operación	Cuota predeterminada	Se puede aumentar
de producto por producto de portal		
Páginas de producto por producto de portal	40	Sí
Tamaño de carga	200 KB	No
Tamaño de la página de documentación por página de punto de conexión REST del producto	32,000 caracteres	No
Tamaño de página personalizado para las páginas de productos	32,000 caracteres	No
Nombres de dominio personalizados por portal	1	No
Autorizadores por portal	1	No

Cuotas para configurar y ejecutar WebSocket en API Gateway

Las siguientes cuotas se aplican a la configuración y la ejecución de una API de WebSocket en Amazon API Gateway. Si la cuota es por API, solo se puede aumentar en API específicas, no en todas las API de una cuenta. Para obtener información sobre las cuotas en el nivel de cuenta, consulte Cuotas y notas importantes

Recurso u operación	Cuota predeterminada	Se puede aumentar
Conexiones nuevas por segundo por cuenta (en todas las API de WebSocket) por región	500	Sí
Conexiones simultáneas	No aplicable *	No aplicable
AWS LambdaAutorizadores de por cada API	10	Sí Para aumentar esta cuota, póngase en contacto con AWS Support Center
AWS LambdaTamaño de resultado de autorizador de	8 KB	No
Rutas por API	300	Sí
Integraciones por API	300	Sí

Recurso u operación	Cuota predeterminada	Se puede aumentar
		Para aumentar esta cuota, póngase en contacto con AWS Support Center
Tiempo de espera de integración	50 milisegundos - 29 segundos para todos los tipos de integraciones, incluidas las integraciones de Lambda, proxy de Lambda, HTTP, proxy de HTTP y AWS.	No
Etapas por API	10	Sí
Tamaño de trama de WebSocket	32 KB	No
Tamaño de carga de mensajes	128 KB **	No
Tamaño máximo de plantilla de asignación	300 KB	No
Duración de la conexión para la API de WebSocket	2 horas	No
Tiempo de inactividad de conexión	10 minutos	No

Recurso u operación	Cuota predeterminada	Se puede aumentar
Longitud, en caracteres, de la URL para una API de WebSocket	4096	No
Tamaño de la plantilla de registro de acceso	3 KB	No

* API Gateway no impone un límite en las conexiones simultáneas. El número máximo de conexiones simultáneas está determinado por la tasa de conexiones nuevas por segundo y la duración máxima de la conexión de dos horas. Por ejemplo, con el límite predeterminado de 500 conexiones nuevas por segundo, si los clientes se conectan a la velocidad máxima durante dos horas, API Gateway puede servir hasta 3 600 000 conexiones simultáneas.

** Debido al límite del tamaño de la trama de WebSocket de 32 KB, los mensajes que superen este límite se deben dividir en varias tramas, cada una con un tamaño máximo de 32 KB. Esto se aplica a los comandos `@connections`. Si se recibe un mensaje (o una trama) de mayor tamaño, la conexión se cierra con el código 1009.

Notas importantes de Amazon API Gateway

En la siguiente sección se ofrece más información que podría afectar al uso de API Gateway.

Temas

- Notas importantes de Amazon API Gateway para las API de HTTP
- Notas importantes de Amazon API Gateway para las API HTTP y de WebSocket
- Notas importantes de Amazon API Gateway para las API REST y de WebSocket
- Notas importantes de Amazon API Gateway para las API de WebSocket
- Notas importantes de Amazon API Gateway para las API REST

Notas importantes de Amazon API Gateway para las API de HTTP

- Las API de HTTP traducen los encabezados `X-Forwarded-*` entrantes como un encabezado `Forwarded` estándar y anexan el host, el protocolo y la IP de salida.
- API Gateway agrega el encabezado Content-type a la solicitud si no hay carga útil y Content-Length es 0.

Notas importantes de Amazon API Gateway para las API HTTP y de WebSocket

- Amazon API Gateway no admite oficialmente Signature Version 4A para las API HTTP y de WebSocket.

Notas importantes de Amazon API Gateway para las API REST y de WebSocket

- API Gateway no permite compartir un nombre de dominio personalizado en las API REST y de WebSocket.
- Los nombres de etapas solo pueden contener caracteres alfanuméricos, guiones y caracteres de subrayado. La longitud máxima es de 128 caracteres.
- Las rutas `/ping` y `/sping` están reservadas para la comprobación de estado del servicio. No se producirá el resultado previsto si se usan estos recursos en el nivel de raíz de la API.
- API Gateway limita actualmente los eventos de registro a 1024 bytes. Los eventos de registros de más de 1024 bytes, como cuerpos de solicitud y respuesta, los truncará API Gateway antes de su envío a CloudWatch Logs.
- En la actualidad, las métricas de CloudWatch limitan los nombres y los valores de las dimensiones a 255 caracteres XML válidos. (Para obtener más información, consulte la guía del usuario de CloudWatch). Los valores de dimensión son una función de los nombres definidos por el usuario, incluido el nombre de API, el nombre de etiqueta (etapa) y el nombre de recurso. Cuando elija estos nombres, asegúrese de no superar los límites de las métricas de CloudWatch.
- El tamaño máximo de una plantilla de asignación es de 300 KB.

Notas importantes de Amazon API Gateway para las API de WebSocket

- API Gateway admite cargas de mensajes de hasta 128 KB con un tamaño máximo de trama de 32 KB. Si un mensaje supera los 32 KB, se debe dividir en varias tramas, cada una con tamaño máximo de 32 KB. Si se recibe un mensaje más grande, la conexión se cierra con el código 1009.

Notas importantes de Amazon API Gateway para las API REST

- El carácter de barra vertical del texto sin formato (`|`) y el carácter de llave (`{` o `}`) no son compatibles con las cadenas de consulta URL y deben codificarse en formato URL.
- El carácter de punto y coma (`;`) no se admite en las cadenas de consulta URL de la solicitud ni en los resultados de los datos que se van a dividir.
- Las API de REST decodifican los parámetros de solicitud codificados con URL antes de pasarlos a las integraciones de backend. Para los parámetros de solicitud de UTF-8, las API de REST decodifican los parámetros y, a continuación, los pasan en formato Unicode a las integraciones de backend. Las API de REST codifican en URL el carácter de porcentaje (`%`) antes de pasarlo a las integraciones de backend.
- Cuando se utiliza la consola de API Gateway para probar una API, es posible que aparezca la respuesta "errores de punto de conexión desconocido" si un certificado autofirmado se presenta en el backend, si falta el certificado intermedio en la cadena de certificados o si el backend genera cualquier otra excepción relacionada con un certificado que no se reconoce.
- En el caso de la entidad `Resource` o `Method` de la API con una integración privada, debe eliminar estas entidades después de eliminar cualquier referencia codificada de forma rígida de `VpcLink`. De lo contrario, quedará una integración pendiente y aparecerá un error en el que se indicará que el enlace VPC sigue en uso aunque la entidad `Resource` o `Method` se haya eliminado. Este comportamiento no se aplica cuando la integración privada hace referencia a `VpcLink` a través de una variable de etapa.
- Los siguientes backends no son compatibles con la autenticación de clientes SSL con API Gateway:
  - NGINX
  - Heroku
- API Gateway admite la mayor parte de la especificación de OpenAPI 2.0 y la especificación de OpenAPI 3.0, con las siguientes excepciones:

- Los segmentos de ruta solo pueden contener caracteres alfanuméricos, guiones, guiones bajos, puntos, comas, dos puntos y llaves. Los parámetros de la ruta deben ser segmentos de ruta separados. Por ejemplo, "resource/{path_parameter_name}" es válido; "resource{path_parameter_name}" no lo es.
- Los nombres de modelo pueden contener únicamente caracteres alfanuméricos.
- Para los parámetros de entrada, solo se admiten los siguientes atributos: , , , y: `name`, `in`, `required`, `type`, `description`. Los demás atributos se ignoran.
- El tipo `securitySchemes`, si se utiliza, debe ser `apiKey`. Sin embargo, OAuth 2 y la autenticación de HTTP Basic se admiten a través de autorizadores de Lambda; la configuración OpenAPI se consigue a través de extensiones de proveedor.
- El campo `deprecated` no es compatible y se descarta en API exportadas.
- Los modelos de API Gateway se definen utilizando esquemas JSON, borrador 4, en lugar de los esquemas JSON que utiliza OpenAPI.
- El parámetro `discriminator` no se admite en ningún objeto de esquema.
- La etiqueta `example` no se admite.
- El campo `nullable` no se admite.
- `exclusiveMinimum` no es compatible con API Gateway.
- Las etiquetas `maxItems` y `minItems` no están incluidas en la validación de solicitud sencilla. Para solucionarlo, actualice el modelo después de la importación antes de efectuar la validación.
- Para OpenAPI 3.0, no puede tener `oneOf`, `anyOf` o `allOf` que utilice `$ref` en una definición dentro del mismo esquema. Puede ingresar directamente el esquema o definir un recurso de modelo de API Gateway independiente. Para obtener más información, consulte the section called “Creación de modelos más complejos”.
- `oneOf` no es compatible con la generación de OpenAPI 2.0 o SDK.
- El campo `readOnly` no se admite.
- `$ref` no puede utilizarse para hacer referencia a otros archivos.
- Las definiciones de respuesta con el formato `"500": {"$ref": "#/responses/UnexpectedError"}` no se admiten en la raíz de documentos de OpenAPI. Para solucionar este problema, sustituya la referencia por el esquema insertado.
- Los números de tipo `Int32` o `Int64` no se admiten. A continuación se muestra un ejemplo:

```
"elementId": {
    "description": "Working Element Id",
```

```
    "format": "int32",
    "type": "number"
}
```

- El tipo de formato de número decimal (`"format": "decimal"`) no se admite en una definición de esquema.
- En las respuestas a métodos, la definición del esquema debe ser un tipo de objeto y no puede tener un tipo primitivo. Por ejemplo, `"schema": { "type": "string"}` no se admite. Sin embargo, puede solucionar este problema con el siguiente tipo de objeto:

```
"schema": {
     "$ref": "#/definitions/StringResponse"
           }

 "definitions": {
    "StringResponse": {
      "type": "string"
    }
  }
```

- API Gateway no utiliza seguridad de nivel raíz definida en la especificación OpenAPI. Por lo tanto hay que definir la seguridad en un nivel de funcionamiento para que se aplique de forma adecuada.
- La `default` palabra clave no ese admite.

- API Gateway establece las siguientes restricciones y limitaciones al administrar métodos con una integración de Lambda o una integración HTTP.
  - Al procesar los nombres de encabezado y los parámetros de consulta, se distingue entre mayúsculas y minúsculas.
  - En la siguiente tabla se enumeran los encabezados que se pueden haber abandonado, reasignado o modificado al enviarse al punto de conexión de integración o desde él. En esta tabla:
    - `Remapped` significa que el nombre del encabezado se ha cambiado de *`<string>`* a `X-Amzn-Remapped-`*`<string>`*.

      `Remapped Overwritten` significa que el nombre del encabezado se cambia de *`<string>`* a `X-Amzn-Remapped-`*`<string>`* y se sobrescribe el valor.

Nombre del encabezado	Solicitud (**http**/**http_proxy** /**lambda**)	Response: (Respuesta (**http**/**http_proxy** /**lambda**)
`Age`	Paso a través	Paso a través
`Accept`	Paso a través	Abandonado/ Paso a través/ Paso a través
`Accept-Charset`	Paso a través	Paso a través
`Accept-Encoding`	Paso a través	Paso a través
`Authorization`	Transferencia directa *	Reasignado
`Connection`	Paso a través/Paso a través/Abandonado	Reasignado
`Content-Encoding`	Paso a través/Abandonado/Paso a través	Paso a través
`Content-Length`	Paso a través (generado basándose en el cuerpo)	Paso a través
`Content-MD5`	Abandonado	Reasignado

Nombre del encabezado	Solicitud (**http**/**http_proxy** /**lambda**)	Response: (Respuesta (**http**/**http_proxy** /**lambda**)
`Content-Type`	Paso a través	Paso a través
`Date`	Paso a través	Reasignado/ Sobrescrito
`Expect`	Abandonado	Abandonado
`Host`	Sobrescrito en el punto de conexión de integración	Abandonado
`Max-Forwards`	Abandonado	Reasignado
`Pragma`	Paso a través	Paso a través
`Proxy-Authenticate`	Abandonado	Abandonado
`Range`	Paso a través	Paso a través
`Referer`	Paso a través	Paso a través
`Server`	Abandonado	Reasignado/ Sobrescrito

Nombre del encabezado	Solicitud (**http**/**http_proxy** /**lambda**)	Response: (Respuesta (**http**/**http_proxy** /**lambda**)
`TE`	Abandonado	Abandonado
`Transfer-Encoding`	Abandonado/Abandonado/Excepción	Abandonado
`Trailer`	Abandonado	Abandonado
`Upgrade`	Abandonado	Abandonado
`User-Agent`	Paso a través	Reasignado
`Via`	Abandonado/Abandonado/Paso a través	Paso a través/ Abandonado/ Abandonado
`Warn`	Paso a través	Paso a través
`WWW-Authenticate`	Abandonado	Reasignado

* El encabezado `Authorization` se elimina si contiene una firma Signature Version 4 o si se usa la autorización de `AWS_IAM`.

- El SDK de Android de una API generado por API Gateway utiliza la clase `java.net.HttpURLConnection`. Esta clase producirá una excepción no administrada en

dispositivos con Android 4.4 y versiones anteriores si la reasignación del encabezado `WWW-Authenticate` a `X-Amzn-Remapped-WWW-Authenticate` produce una respuesta 401.

- A diferencia de los SDK de Java, Android e iOS de una API generados por API Gateway, el SDK de JavaScript de una API generado por API Gateway no admite reintentos de error de nivel 500.
- La invocación de prueba de un método utiliza el tipo de contenido predeterminado de `application/json` y no tiene en cuenta las especificaciones de cualquier otro tipo de contenido.
- Al enviar solicitudes a una API pasando el encabezado `X-HTTP-Method-Override`, API Gateway anula el método. Por lo tanto, con el fin de pasar el encabezado al backend, el encabezado tiene que añadirse a la solicitud de integración.
- Cuando una solicitud contiene varios tipos de medios en su encabezado `Accept`, API Gateway solo respeta el primer tipo de medio `Accept`. En aquella situación en la que no pueda controlar el orden de los tipos de medios `Accept` y el tipo de medio del contenido binario no sea el primero de la lista, puede agregar el primer tipo de medio `Accept` en la lista `binaryMediaTypes` de la API; API Gateway devolverá su contenido como binario. Por ejemplo, para enviar un archivo JPEG utilizando un elemento `<img>` en un navegador, el navegador puede enviar `Accept:image/webp,image/*,*/*;q=0.8` en una solicitud. Al añadir `image/webp` a la lista `binaryMediaTypes`, el punto de conexión recibirá el archivo JPEG como binario.
- Actualmente, no se permite personalizar la respuesta predeterminada de la gateway en `413 REQUEST_TOO_LARGE`.
- API Gateway incluye un encabezado `Content-Type` para todas las respuestas de integración. De forma predeterminada, el tipo de contenido es `application/json`.

Historial de revisión

En la siguiente tabla se describen los cambios importantes que se han realizado en la documentación desde la última versión de Amazon API Gateway. Para recibir notificaciones sobre actualizaciones de esta documentación, suscríbase a una fuente RSS eligiendo el botón RSS en el panel del menú de arriba.

- Última actualización de la documentación: 02 de diciembre de 2025

Cambio	Descripción	Fecha
Agregación de una API de REST de API Gateway como destino para la puerta de enlace de Amazon Bedrock AgentCore	API Gateway ahora admite la agregación de una API de REST como destino para la puerta de enlace de Amazon Bedrock AgentCore. Para obtener más información, consulte Agregación de una API de REST de API Gateway como destino para la puerta de enlace de Amazon Bedrock AgentCore.	2 de diciembre de 2025
Integración privada con equilibrador de carga de aplicación	API Gateway ahora admite integraciones privadas con equilibradores de carga de aplicación para API de REST. Para obtener más informaci ón, consulte Integraciones privadas para API de REST.	21 de noviembre de 2025
Políticas de seguridad para API de REST y nombres de dominio personalizados	API Gateway ahora admite políticas de seguridad de TLS mejoradas en API de REST y nombres de dominio personali zados. Para obtener más	19 de noviembre de 2025

	información, consulte Políticas de seguridad para API de REST en API Gateway.	
Transmisión de respuestas para API de REST	Las API de REST de API Gateway ahora admiten la transmisión progresiva de cargas útiles de respuesta a los clientes a medida que están disponibles. Para obtener más información, consulte Transmisión de la respuesta de integración para las integraciones de proxy.	19 de noviembre de 2025
Portales para desarrolladores	API Gateway ahora es compatible con los portales, que son una ubicación centralizada para que los clientes detecten y prueben las API y los productos del portal que representan un servicio o una funcionalidad que desea compartir. Para obtener más información, consulte Portales de API Gateway.	19 de noviembre de 2025
SIGv4a para API de REST	API Gateway ahora admite el uso de Signature Version 4a para API de REST. Para obtener más información, consulte Modelo de permisos de API Gateway para invocar una API.	19 de agosto de 2025

Reglas de enrutamiento para las API de REST	API Gateway admite ahora reglas de enrutamiento para las API de REST que utilizan nombres de dominio personalizados. Puede enrutar dinámicamente las solicitud es entrantes basándose en valores de encabezado HTTP, rutas base de URL o una combinación de ambos. Para obtener más información, consulte Envío de tráfico a las API a través del nombre de dominio personalizado.	3 de junio de 2025
Se ha agregado compatibi lidad para puntos de conexión de doble pila (IPv4 e IPv	API Gateway ahora admite puntos de conexión de doble pila para las API de REST, HTTP y de WebSocket, y los nombres de dominio personali zados. Para obtener más información, consulte tipos de direcciones IP.	28 de marzo de 2025
Nombres de dominio personali zados para API privadas	Los nombres de dominio personalizados no son compatibles con las API privadas. Para obtener más información, consulte Nombres de dominio personali zados para API privadas.	21 de noviembre de 2024

Se ha agregado compatibilidad para TLS 1.3	API Gateway ahora es compatible con TLS 1.3 en las API de REST regionales, las API HTTP y las API de WebSocket. Para obtener más información, consulte Elección de una política de seguridad para el dominio personalizado en API Gateway.	15 de febrero de 2024
Actualizaciones de la consola de la API de REST y API de WebSocket	Información de la consola actualizada para las API de REST y las API de WebSocket.	10 de diciembre de 2023
Actualización de la documentación	Se ha actualizado la información conceptual y se han creado nuevos tutoriales sobre transformaciones de datos y temas de validación de solicitudes para las API de REST de API Gateway. Para obtener más información, consulte Uso de la validación de solicitudes en API Gateway y Configuración de transformaciones de datos para API de REST.	22 de junio de 2023

Configuración de la conmutación por error de DNS para API Gateway de varias regiones	Se ha agregado compatibilidad para utilizar las comprobaciones de estado de Amazon Route 53 para controlar la conmutación por error de DNS de una API de REST de API Gateway en una Región de AWS principal a una en una región secundaria. Para obtener más información, consulte Configurar las comprobaciones de estado personalizadas para la conmutación por error de DNS.	31 de octubre de 2022
Actualización de la documentación	Se han actualizado los resúmenes de las características principales de las API de REST y API HTTP. Para obtener más información, consulte Elección entre API de REST y API HTTP.	31 de mayo de 2022
Actualización de la política administrada	Se ha añadido soporte `acm:GetCertificate` a la política de `AWSServiceRoleForAPIGateway` . Para obtener más información, consulte Uso de roles vinculados a servicios para API Gateway.	12 de julio de 2021

Mapeo de parámetros para las API HTTP	Se agregó soporte para el mapeo de parámetros para las API HTTP. Para obtener más información, consulte Transformación de solicitudes y respuestas de API.	7 de enero de 2021
Desactivación del punto de conexión predeterminado para una API de REST	Se ha agregado compatibilidad para desactivar el punto de conexión predeterminado para las API de REST. Para obtener más información, consulte Desactivación del punto de conexión predeterminado para una API de REST.	29 de octubre de 2020
Autenticación TLS mutua	Se ha agregado compatibilidad para la autenticación TLS mutua para las API de REST y las API HTTP. Para obtener más información, consulte Configuración de la autenticación TLS mutua para una API de REST y Configuración de la autenticación TLS mutua para una API HTTP.	17 de septiembre de 2020
Autorizadores de AWS Lambda para API HTTP	Se ha agregado compatibilidad con los autorizadores de AWS Lambda para las API HTTP. Para obtener más información, consulte Uso de autorizadores de AWS Lambda para API HTTP.	9 de septiembre de 2020

Integraciones de servicios de AWS con API HTTP	Se ha agregado compatibilidad con las integraciones de servicios de AWS para las API HTTP. Para obtener más información, consulte Uso de integraciones de servicios de AWS para API HTTP.	20 de agosto de 2020
Dominios personalizados comodín para API HTTP	Se ha agregado compatibilidad con nombres de dominio personalizados comodín para las API HTTP. Para obtener más información, consulte Nombres de dominio personalizados comodín.	10 de agosto de 2020
Mejoras del portal para desarrolladores sin servidor	Se ha añadido la administración de usuarios al panel del administrador y soporte para exportar definiciones de la API. Para obtener más información, consulte Usar el portal para desarrolladores de aplicaciones sin servidor para catalogar las API de API Gateway.	25 de junio de 2020
Compatibilidad con `Sec-WebSocket-Protocol` de API de WebSocket	Se ha añadido compatibilidad con el campo `Sec-WebSocket-Protocol`. Para obtener más información, consulte Configuración de una ruta $connect que requiere un subprotocolo WebSocket.	16 de junio de 2020

Exportación de una API HTTP	Se ha agregado compatibilidad para exportar las definiciones de OpenAPI 3.0 de una API HTTP. Para obtener más información, consulte Exportación de una API HTTP desde API Gateway.	20 de abril de 2020
Documentación de seguridad	Documentación de seguridad añadida. Para obtener más información, consulte Seguridad en Amazon API Gateway.	31 de marzo de 2020
Documentación reorganizada	Se ha reorganizado la guía para desarrolladores.	12 de marzo de 2020
Disponibilidad general de la API HTTP	Se han publicado las API HTTP en disponibilidad general. Para obtener más información, consulte Trabajar con API HTTP.	12 de marzo de 2020
Registro de API HTTP	Se ha añadido soporte para `$context.integrationErrorMessage` en los registros de API HTTP. Para obtener más información, consulte Variables de registro de API HTTP.	26 de febrero de 2020

AWS Variables de para la importación de OpenAPI	Se ha añadido compatibilidad para las variables de AWS en las definiciones de OpenAPI. Para obtener más informaci ón, consulte Variables de AWS para la importación de OpenAPI.	17 de febrero de 2020
API HTTP	Se han lanzado las API HTTP en versión beta. Para obtener más información, consulte API HTTP.	4 de diciembre de 2019
Nombres de dominio personali zados comodín	Se ha añadido compatibi lidad con nombres de dominio personalizados comodín. Para obtener más información, consulte Nombres de dominio personalizados comodín.	21 de octubre de 2019
Registro de Amazon Data Firehose	Se ha agregado compatibi lidad para Amazon Data Firehose como destino para los datos de registro de acceso. Para obtener más información, consulte Uso de Amazon Data Firehose como destino para el registro de acceso de API Gateway.	15 de octubre de 2019

Alias de Route53 para llamar a las API privadas	Se ha añadido compatibilidad con los registros de DNS de alias de Route53 adicionales para llamar a las API privadas. Para obtener más informaci ón, consulte Acceso a la API privada a través de un alias de Route53.	18 de septiembre de 2019
Control de acceso basado en etiquetas para API WebSocket	Se ha añadido compatibi lidad con control de acceso basado en etiquetas para API WebSocket. Para obtener más información, consulte Recursos de API Gateway que se pueden etiquetar.	27 de junio de 2019
Selección de versión de TLS para dominios personalizados	Se ha añadido compatibilidad para la selección de versión de Transport Layer Security (TLS) para las API que se implementan en dominios personalizados. Consulte la nota en Elegir una versión mínima de TLS para un dominio personalizado en API Gateway.	20 de junio de 2019

Políticas de punto de conexión de VPC para API privadas	Se ha agregado compatibilidad para mejorar la seguridad de API privadas, asociando políticas de punto de conexión a los puntos de conexión de VPC de interfaz. Para obtener más información, consulte Utilizar políticas de punto de conexión de VPC para API privadas en API Gateway.	4 de junio de 2019
Documentación actualizada	Se ha reescrito Introducción a Amazon API Gateway. Se han movido tutoriales a Tutoriales de Amazon API Gateway.	29 de mayo de 2019
Control de acceso basado en etiquetas para las API de REST	Se ha añadido compatibilidad para control de acceso basado en etiquetas para las API de REST. Para obtener más información, consulte Uso de etiquetas con las políticas de IAM para controlar el acceso a los recursos de API Gateway.	23 de mayo de 2019

Documentación actualizada	Se han reescrito seis temas: ¿Qué es Amazon API Gateway?, Tutorial: Desarrollo de una API con la integración de proxy HTTP, Tutorial: Creación de una API de REST de Calc con tres integraciones que no sean de proxy, Plantilla de mapeo de API Gateway y referencia de la variable de registro de acceso, Uso de autorizadores de Lambda de API Gateway y Habilitar CORS para un recurso de API de REST de API Gateway.	5 de abril de 2019
Mejoras del portal para desarrolladores sin servidor	Se han agregado el panel de administrador y otras mejoras para facilitar la publicación de las API en el portal para desarrolladores de Amazon API Gateway. Para obtener más información, consulte Uso del portal para desarrolladores para catalogar las API.	28 de marzo de 2019
Compatibilidad con AWS Config	Se ha agregado compatibilidad con AWS Config. Para obtener más información, consulte Monitoreo de la configuración de la API de API Gateway con AWS Config.	20 de marzo de 2019

Compatibilidad con CloudFormation	Se agregó la API de API Gateway V2 a la referencia de la plantilla de CloudFormation. Para obtener más información, consulte Referencia de tipos de recursos de Amazon API Gateway V2.	7 de febrero de 2019
Compatibilidad con las API de WebSocket	Se ha agregado compatibilidad con las API de WebSocket. Para obtener más información, consulte Creación de una API de WebSocket en Amazon API Gateway.	18 de diciembre de 2018
Portal para desarrolladores sin servidor disponible a través d AWS Serverless Application Repository	La aplicación del portal para desarrolladores sin servidor de Amazon API Gateway ahora está disponible desde AWS Serverless Application Repository (además de GitHub). Para obtener más información, consulte Uso del portal para desarrolladores para catalogar las API de API Gateway.	16 de noviembre de 2018
Compatibilidad con AWS WAF	Se ha agregado compatibilidad con AWS WAF (Web Application Firewall). Para obtener más información, consulte Control de acceso a una API mediante AWS WAF.	5 de noviembre de 2018

Portal para desarrolladores de aplicaciones sin servidor	Amazon API Gateway ahora ofrece un portal para desarrolladores totalmente personalizable como una aplicación sin servidor que puede implementar para publicar sus API de API Gateway. Para obtener más información, consulte Uso del portal para desarrolladores para catalogar las API de API Gateway.	29 de octubre de 2018
Compatibilidad con encabezados de varios valores y parámetros de cadenas de consulta	Amazon API Gateway ahora es compatible con varios encabezados y parámetros de cadena de consulta que tengan el mismo nombre. Para obtener más información, consulte Compatibilidad con encabezados de varios valores y parámetros de cadena de consulta.	4 de octubre de 2018
Compatibilidad con OpenAPI	Amazon API Gateway ahora es compatible con OpenAPI 3.0 y OpenAPI (Swagger) 2.0.	27 de septiembre de 2018
Documentación actualizada	Se ha agregado un nuevo tema: Cómo afectan las políticas de recursos de Amazon API Gateway al flujo de trabajo de autorización.	27 de septiembre de 2018

Integración activa de AWS X-Ray	A partir de ahora, puede utilizar AWS X-Ray para rastrear y analizar las latencias en las solicitudes de los usuarios a medida que avanzan a través de las API para los servicios subyacentes. Para obtener más información, consulte Rastreo de la ejecución de la API de API Gateway con AWS X-Ray.	6 de septiembre de 2018
Mejoras en el almacenamiento en caché	Solo los métodos `GET` tendrán el almacenamiento en caché habilitado de forma predeterminada cuando habilite el almacenamiento en caché para una etapa de la API. Esto ayuda a garantizar la seguridad de la API. Puede habilitar el almacenamiento en caché para otros métodos invalidando la configuración del método. Para obtener más información, consulte Habilitar el almacenamiento en caché de la API para mejorar la capacidad de respuesta.	20 de agosto de 2018

Revisión de Service limits	Se han revisado varios límites: mayor número de API por cuenta. Se han aumentado los límites de tasa de las API Create/Import/Deploy. Se han corregido algunas tasas, que pasan de ser por minuto a ser por segundo. Para obtener más información, consulte Límites.	13 de julio de 2018
Invalidación de encabezados y parámetros de solicitud y respuesta de API	Se ha agregado la posibilidad de invalidar encabezados de solicitud, cadenas de consulta y rutas de acceso, así como de encabezados de respuesta y códigos de estado. Para obtener más información, consulte Uso de una plantilla de mapeo para invalidar códigos de estado y parámetros de solicitud y de respuesta de una API.	12 de julio de 2018

Limitación controlada de nivel de método para planes de uso	Se ha agregado la posibilidad de configurar limitaciones controladas predeterminadas por método, así como limitaciones controladas para métodos individuales de API en el plan de uso. Esta configuración se suma a las limitaciones controladas de nivel de cuenta y predeterminadas de nivel de método ya existentes que se pueden definir en la configuración de etapas. Para obtener más información, consulte Limitar las solicitudes de la API para mejorar el rendimiento.	11 de julio de 2018
Las notificaciones de actualización de la guía para desarrolladores de API Gateway ya están disponibles a través de RSS	La versión HTML de la guía para desarrolladores de API Gateway ahora admite una fuente RSS para las actualizaciones que se documentan en esta página Historial de revisión. La fuente RSS incluye las actualizaciones realizadas el 27 de junio de 2018 y después de esa fecha. Las actualizaciones anunciadas anteriormente siguen estando disponibles en esta página. Utilice el botón RSS del panel del menú superior para suscribirse a la fuente.	27 de junio de 2018

Actualizaciones anteriores

En la siguiente tabla se describen los cambios importantes que se han realizado en cada una de las versiones de la guía para desarrolladores de API Gateway anteriores al 27 de junio de 2018.

Cambio	Descripción	Fecha de modificación
API privadas	Se ha agregado compatibilidad con las API privadas que se exponen a través de los puntos de conexión de VPC de la interfaz. El tráfico dirigido a la API privada no sale de la red de Amazon; está aislado de la red pública de Internet.	14 de junio de 2018
Autorizadores e integraciones de Lambda entre cuentas y autorizadores de grupos de usuarios de Amazon Cognito entre cuentas	Utilice una función de AWS Lambda de otra cuenta de AWS como una función de autorizador de Lambda o como un backend de integración de la API. O bien, utilice un grupo de usuarios de Amazon Cognito como autorizador. La otra cuenta puede estar en cualquier región donde Amazon API Gateway esté disponible. Para obtener más información, consulte the section called “Configuración de un autorizador de Lambda entre cuentas”, the section called “Tutorial: Creación de una API de REST con integración de proxy de Lambda entre cuentas” y the section called “Configuración de un autorizador de Amazon Cognito entre cuentas para una API REST”.	2 de abril de 2018
Políticas de recursos para las API	Utilice políticas de recursos de API Gateway para permitir que los usuarios de otra cuenta de AWS obtengan acceso seguro a la API o para permitir que se invoque la API únicamente desde determinados bloques de CIDR o rangos de direcciones IP de origen. Para obtener más información, consulte the section called “Uso de políticas de recursos de API Gateway”.	2 de abril de 2018
Etiquetado de recursos de API Gateway	Etiquete una etapa de API con hasta 50 etiquetas para asignar los costos a las solicitudes de API y el almacenamiento en caché de API Gateway. Para obtener más	19 de diciembre de 2017

Cambio	Descripción	Fecha de modificación
	información, consulte the section called “Configuración de etiquetas”.	
Compresión y descompresión de cargas	Permita que la API pueda invocarse con cargas comprimidas utilizando una de las codificaciones de contenido compatibles. Las cargas comprimidas se someterán a un proceso de asignación si se especifica una plantilla de asignación de cuerpo. Para obtener más información, consulte the section called “Codificación de contenido”.	19 de diciembre de 2017
Clave de API procedente de un autorizador personalizado	Devuelva a API Gateway una clave de API procedente de un autorizador personalizado para aplicar un plan de uso de los métodos de API que necesitan la clave. Para obtener más información, consulte the section called “Elección de un origen de clave de API en API Gateway”.	19 de diciembre de 2017
Autorización con ámbitos de OAuth 2	Habilite la invocación de un método de autorización utilizando ámbitos de OAuth 2 con el autorizador `COGNITO_USER_POOLS`. Para obtener más información, consulte the section called “Uso de grupos de usuarios de Amazon Cognito como autorizador para una API REST”.	14 de diciembre de 2017
Integración privada y conexión VPC	Cree una API con la integración privada de API Gateway para proporcionar a los clientes acceso a los recursos HTTP/HTTPS de Amazon VPC desde fuera de dicha VPC a través de un recurso VpcLink. Para obtener más información, consulte the section called “Tutorial: Creación de una API de REST con una integración privada” y the section called “Configuración de una integración privada”.	30 de noviembre de 2017

Cambio	Descripción	Fecha de modificación
Implemente una versión Canary para pruebas de API	Agregue una versión Canary a una implementación de API existente para probar la última versión de la API mientras que la versión actual se mantiene operativa en la misma etapa. Puede definir un porcentaje del tráfico de la etapa para la versión Canary y especificar que la ejecución y el acceso de esta versión se incluyan en registros de CloudWatch Logs independientes. Para obtener más información, consulte the section called “Configuración de una implementación de versión canario”.	28 de noviembre de 2017
Registro de acceso	Registre el acceso de los clientes a la API con los datos procedentes de variables $context en el formato que elija. Para obtener más información, consulte the section called “CloudWatch Logs”.	21 de noviembre de 2017
SDK de Ruby para una API	Cree un SDK de Ruby para una API y utilícelo para invocar los métodos de API. Para obtener más información, consulte the section called “Generación del SDK de Ruby de una API en API Gateway” y the section called “Uso de un SDK de Ruby generado por API Gateway para una API REST”.	20 de noviembre de 2017
Punto de conexión de API regional	Especifique un punto de conexión de API regional para crear una API para los clientes no móviles. Un cliente no móvil, como una instancia EC2, se ejecuta en la misma región de AWS donde se ha implementado la API. Al igual que ocurre con la API optimizada para límites, puede crear un nombre de dominio personalizado para una API regional. Para obtener más información, consulte the section called “Tipos de puntos de conexión de API Gateway” y the section called “Configuración de un nombre de dominio personalizado regional”.	2 de noviembre de 2017

Cambio	Descripción	Fecha de modificación
Autorizador de solicitud personalizado	Utilice el autorizador de solicitud personalizado para suministrar información de la autenticación del usuario en los parámetros de solicitud para autorizar llamadas al método de la API. Los parámetros de solicitud incluyen encabezados y parámetros de cadenas de consulta, así como variables de etapa y de contexto. Para obtener más información, consulte Uso de autorizadores Lambda de API Gateway.	15 de septiembre de 2017
Personalización de las respuestas de la gateway	Personalice las respuestas de gateway generadas por API Gateway a las solicitudes API que no consiguieron llegar al backend de integración. Un mensaje de gateway personalizado puede proporcionar al intermediario mensajes de error personalizados específicos de la API, incluidos los encabezados de CORS necesarios, o puede transformar los datos de respuesta de gateway en un formato de un intercambio externo. Para obtener más información, consulte Configuración de respuestas de gateway para personalizar respuestas de errores.	6 de junio de 2017
Mapeo de propiedades de errores personalizados de Lambda con encabezados de respuesta de método	Asigne las propiedades de errores personalizados devueltas por Lambda a los parámetros de encabezado de respuesta de método mediante el parámetro `integration.response.body` , utilizando API Gateway para deserializar el objeto de error personalizado representado en forma de cadena en tiempo de ejecución. Para obtener más información, consulte Gestionar los errores de Lambda personalizados en API Gateway.	6 de junio de 2017

Cambio	Descripción	Fecha de modificación
Aumento de los límites de solicitudes	Aumente el límite de solicitudes de estado estable de nivel de cuenta a 10 000 solicitudes por segundo (sps) y el límite de ráfaga a 5000 solicitudes simultáneas. Para obtener más información, consulte Limitación de las solicitudes a sus API de REST para mejorar el rendimiento en API Gateway.	6 de junio de 2017
Validación de solicitudes de método	Configure validadores de solicitudes básicos en el nivel de API o de método para que API Gateway pueda validar las solicitudes entrantes. API Gateway verifica que los parámetros obligatorios están establecidos y no están vacíos, y verifica que el formato de las cargas aplicables es compatible con el modelo configurado. Para obtener más información, consulte Solicitud de la validación de las API de REST en API Gateway.	11 de abril de 2017
Integración con ACM	Utilice certificados de ACM para los nombres de dominio personalizados de la API. Puede crear un certificado en AWS Gestor de certificados o importar un certificado con formato PEM existente en ACM. A continuación, puede hacer referencia al ARN del certificado cuando configure nombre de dominio personalizado para las API. Para obtener más información, consulte Nombre de dominio personalizado para las API de REST públicas en API Gateway.	9 de marzo de 2017
Generar y llamar a un SDK de Java de una API	Deje que API Gateway genere el SDK de Java para su API y utilice el SDK para llamar a la API en su cliente Java. Para obtener más información, consulte Uso de un SDK de Java generado por API Gateway para una API REST.	13 de enero de 2017

Cambio	Descripción	Fecha de modificación
Integración con AWS Marketplace	Venda la API en un plan de uso como un producto SaaS mediante AWS Marketplace. Utilice AWS Marketplace para ampliar el alcance de la API. Use AWS Marketplace para facturar a los clientes en su nombre. Deje que API Gateway gestione la autorización del usuario y la medición del uso. Para obtener más información, consulte Vender sus API como SaaS.	1 de diciembre de 2016
Habilitar la ayuda de documentación para su API	Agregue documentación para las entidades de API en recursos DocumentationPart en API Gateway. Asocie una instantánea de las instancias de `DocumentationPart` a una etapa de API para crear un elemento DocumentationVersion. Publique la documentación de la API exportando una versión de documentación a un archivo externo, como un archivo Swagger. Para obtener más información, consulte Documentación para las API de REST en API Gateway.	1 de diciembre de 2016
Autorizador personalizado actualizado	Una función de Lambda del autorizador de clientes devuelve ahora el identificador principal del intermediario. La función también puede devolver otra información como pares de clave-valor del mapa `context` y una política de IAM. Para obtener más información, consulte Salida de un autorizador de Lambda de API Gateway.	1 de diciembre de 2016
Compatibilidad con cargas binarias	Establezca binaryMediaTypes en la API para admitir cargas binarias de una solicitud o una respuesta. Establezca la propiedad `contentHandling` en un recurso Integration o IntegrationResponse para especificar si la carga binaria se trata como el blob binario nativo, como una cadena codificada en Base64 o como un acceso directo sin modificaciones. Para obtener más información, consulte Tipos de medios binarios para las API de REST en API Gateway.	17 de noviembre de 2016

Cambio	Descripción	Fecha de modificación
Habilitar una integración de proxy con un backend HTTP o Lambda a través de un recurso de proxy de una API.	Cree un recurso de proxy con un parámetro de ruta expansiva con el formato `{proxy+}` y el método catch-all `ANY`. El recurso de proxy se integra con un backend HTTP o Lambda utilizando la integración de proxy HTTP o Lambda, respectivamente. Para obtener más información, consulte Configuración de una integración de proxy con un recurso de proxy.	20 de septiembre de 2016
Ampliar las API seleccionadas en API Gateway como ofertas de productos para sus clientes proporcio nando uno o varios planes de uso	Cree un plan de uso en API Gateway para permitir a los clientes API seleccionados tener acceso a las etapas de API especificadas con las tarifas y cuotas acordadas. Para obtener más información, consulte Planes de uso y clave de API para las API de REST en API Gateway.	11 de agosto de 2016
Habilitación de la autorización en el nivel de método con un grupo de usuarios en Amazon Cognito	Cree un grupo de usuarios en Amazon Cognito y úselo como su propio proveedor de identidades. Puede configura r el grupo de usuarios como un autorizador de nivel de método para conceder acceso a los usuarios que están registrados en el grupo de usuarios. Para obtener más información, consulte Control del acceso a las API de REST con grupos de usuarios de Amazon Cognito como autorizador.	28 de julio de 2016
Habilitar las métricas y dimensiones de Amazon CloudWatc h bajo el espacio de nombres de `AWS/ ApiGateway` .	Las métricas de API Gateway ahora se han estandarizado bajo el espacio de nombres CloudWatch de `AWS/ApiGa teway` . Puede verlos tanto en la consola de API Gateway como en la consola de Amazon CloudWatch. Para obtener más información, consulte Dimensiones y métricas de Amazon API Gateway.	28 de julio de 2016

Cambio	Descripción	Fecha de modificación
Habilitar la rotación de certificados para un nombre de dominio personalizado	La rotación de certificados le permite cargar y renovar un certificado que va a vencer para un nombre de dominio personalizado. Para obtener más información, consulte Rotación de un certificado importado en ACM.	27 de abril de 2016
Documentar los cambios de la consola de Amazon API Gateway actualizada	Obtenga información sobre cómo crear y configurar una API mediante la consola de API Gateway actualizada. Para obtener más información, consulte Tutorial: Crear una API de REST importando un ejemplo y Tutorial: Creación de una API de REST con integración no de proxy HTTP.	5 de abril de 2016
Habilitar la característica Import API para crear una API nueva o actualizar una existente desde definiciones de API externas	Con las características de Import API, puede crear una nueva API o actualizar una existente cargando una definición de API externa expresada en Swagger 2.0 con las extensiones de API Gateway. Para obtener más información sobre Import API, consulte Desarrollo de API de REST mediante OpenAPI en API Gateway.	5 de abril de 2016
Exponer la variable `$input.body` para tener acceso a la carga sin formato como una cadena y la función `$util.parseJson()` para convertir una cadena JSON en un objeto JSON en una plantilla de asignación	Para obtener más información sobre `$input.body` y `$util.parseJson()`, consulte Variables para transformaciones de datos para API Gateway.	5 de abril de 2016

Cambio	Descripción	Fecha de modificación
Habilitar solicitudes cliente con invalidación de caché de nivel de método y mejorar la administración de limitaciones de solicitudes	Vacíe la caché de nivel de etapa de API e invalide entradas de caché individuales. Para obtener más información, consulte Vaciar la caché de etapa de API en API Gateway y Invalidar una entrada de caché de API Gateway. Mejore la experiencia de la consola para la administración de limitaciones de solicitudes de API. Para obtener más información, consulte Limitación de las solicitudes a sus API de REST para mejorar el rendimiento en API Gateway.	25 de marzo de 2016
Habilitación y llamada a la API de API Gateway con autorización personalizada	Cree y configure una función de AWS Lambda para implementar la autorización personalizada. La función devuelve un documento de política de IAM que concede los permisos Allow o Deny a las solicitudes cliente de una API de API Gateway. Para obtener más información, consulte Uso de autorizadores Lambda de API Gateway.	11 de febrero de 2016
Importación y exportación de API de API Gateway con un archivo de definición y extensiones de Swagger	Cree y actualice la API de API Gateway con la especificación de Swagger con las extensiones de API Gateway. Importe las definiciones de Swagger utilizando API Gateway Importer. Exporte una API de API Gateway a un archivo de definición de Swagger mediante la consola de API Gateway o la API de exportación de API Gateway. Para obtener más información, consulte Desarrollo de API de REST mediante OpenAPI en API Gateway y Exportación de una API REST desde API Gateway.	18 de diciembre de 2015
Asignar campos JSON de cuerpo o cuerpo de solicitud o respuesta a parámetros de solicitud o respuesta	Asigne el cuerpo de solicitud de método o sus campos JSON a la ruta, cadena de consulta o encabezados de una solicitud de integración. Asigne el cuerpo de una respuesta de integración o sus campos JSON a encabezados de una respuesta de solicitud. Para obtener más información, consulte Ejemplos de asignación de parámetros para las API de REST en API Gateway.	18 de diciembre de 2015

Cambio	Descripción	Fecha de modificación
Trabajar con variables de etapa en Amazon API Gateway	Aprenda a asociar atributos de configuración a una etapa de implementación de una API en Amazon API Gateway. Para obtener más información, consulte Uso de variables de etapa para una API de REST en API Gateway.	5 de noviembre de 2015
Cómo: Habilitar CORS para un método	Ahora es más fácil habilitar el uso compartido de recursos entre orígenes (CORS) para métodos en Amazon API Gateway. Para obtener más información, consulte CORS para las API de REST en API Gateway.	3 de noviembre de 2015
Cómo: Uso de la autenticación SSL del lado cliente	Utilice Amazon API Gateway para generar certificados SSL que pueda utilizar para autenticar las llamadas a su backend HTTP. Para obtener más información, consulte Uso de un certificado generado por API Gateway para la autenticación de backend en API Gateway.	22 de septiembre de 2015
Integración simulada de métodos	Aprenda a integrar una API de forma simulada con Amazon API Gateway. Esta característica permite a los desarrolladores generar respuestas de API desde API Gateway directamente sin la necesidad de definir previamente un backend de integración final.	1 de septiembre de 2015
Compatibilidad con identidades de Amazon Cognito	Amazon API Gateway ha ampliado el alcance de la variable `$context` para que ahora devuelva información sobre identidades de Amazon Cognito cuando las solicitud es van firmadas con credenciales de Amazon Cognito. Además, hemos añadido una variable `$util` para incluir caracteres en secuencias de escape en JavaScript y codificar direcciones URL y cadenas. Para obtener más información, consulte Variables para transformaciones de datos para API Gateway.	28 de agosto de 2015

Cambio	Descripción	Fecha de modificación
Integración de Swagger	Utilice la herramienta de importación de Swagger en GitHub para importar definiciones de API de Swagger en Amazon API Gateway. Obtenga más información acerca de Extensiones de OpenAPI para API Gateway para crear e implementar API y métodos utilizando la herramienta de importación. Con la herramienta de importación de Swagger también puede actualizar las API existentes.	21 de julio de 2015
Referencia de la plantilla de asignación	Obtenga información sobre el parámetro `$input` y sus funciones en Variables para transformaciones de datos para API Gateway.	18 de julio de 2015
Versión pública inicial	Esta es la versión pública inicial de la guía para desarroll adores de API Gateway.	9 de julio de 2015