



Guia do desenvolvedor

Amazon API Gateway



Amazon API Gateway: Guia do desenvolvedor

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

As marcas comerciais e imagens de marcas da Amazon não podem ser usadas no contexto de nenhum produto ou serviço que não seja da Amazon, nem de qualquer maneira que possa gerar confusão entre os clientes ou que deprecie ou desprestige a Amazon. Todas as outras marcas comerciais que não pertencem à Amazon pertencem a seus respectivos proprietários, que podem ou não ser afiliados, patrocinados pela Amazon ou ter conexão com ela.

Table of Contents

O que é o Amazon API Gateway?	1
Arquitetura do API Gateway	2
Recursos do API Gateway	3
Casos de uso do API Gateway	3
Usar o API Gateway para criar APIs REST	4
Usar o API Gateway para criar APIs HTTP	5
Usar o API Gateway para criar APIs do WebSocket	5
Quem usa o API Gateway?	6
Acessar o API Gateway	7
Parte da infraestrutura sem servidor da AWS	7
Saiba como começar a usar o Amazon API Gateway	8
Conceitos do API Gateway	8
Escolher entre APIs REST e APIs HTTP	14
Tipo de endpoint	15
Segurança	15
Autorização	15
Gerenciamento de APIs	16
Desenvolvimento	17
Monitoramento	17
Integrações	18
Começar a usar o console da API REST	19
Etapa 1: Criar uma função do Lambda	20
Etapa 2: criar uma API REST	20
Etapa 3: criar uma integração de proxy do Lambda	21
Etapa 4: implantar a API	21
Etapa 5: invocar a API	22
Etapa 6 (opcional): limpar	23
Pré-requisitos	25
Inscrever-se para uma Conta da AWS	25
Criar um usuário com acesso administrativo	25
Conceitos básicos	28
Etapa 1: Criar uma função do Lambda	29
Etapa 2: criar uma API HTTP	29
Etapa 3: testar sua API	30

(Opcional) Etapa 4: Limpar	31
Próximas etapas	32
Tutoriais e workshops	34
Tutoriais da API REST	35
Escolher um tutorial de integração do Lambda	35
Tutorial: Criar uma API REST importando um exemplo	60
Escolher um tutorial de integração HTTP	69
Tutorial: Crie uma API REST com uma integração privada	83
Tutorial: Crie uma API REST com uma integração da AWS	89
Tutorial: Crie uma API REST de calculadora com três integrações	95
Tutorial: Crie uma API REST como um proxy do Amazon S3	122
Tutorial: Criar uma API REST como um proxy do Amazon Kinesis	167
Tutorial: como criar uma API REST usando SDKs da AWS ou a AWS CLI	213
Tutorial: Crie uma API REST privada	247
Tutoriais da API HTTP	254
Tutorial: Crie uma API HTTP CRUD com o Lambda e o DynamoDB	254
Tutorial: Crie uma API HTTP com uma integração privada ao Amazon ECS	266
Tutoriais da API WebSocket	273
Tutorial: Crie uma aplicação de bate-papo de WebSocket	274
Tutorial: Crie uma API de WebSocket com uma integração da AWS	279
APIs REST do API Gateway	292
Desenvolver	292
Exemplo de recurso para uma API REST	293
Recursos adicionais da API REST para desenvolvimento	296
Tipos de endpoint do API Gateway	298
Políticas de segurança	302
Tipos de endereço IP para APIs REST no API Gateway	323
Métodos	325
Controle de acesso	346
Integrações	434
Validação da solicitação	529
Transformações de dados	560
Respostas do gateway	645
CORS	658
Tipos de mídia binária	672
Invoke	705

OpenAPI	737
Publicar	752
Implantar APIs REST	753
Nomes de domínios personalizados	800
Otimizar	875
Configurações de cache	875
Codificação de conteúdo	889
Distribuir	896
Planos de uso	896
Documentação de API	924
Geração de SDKs	990
Vender suas APIs como SaaS	1017
Proteger	1021
TLS mútuo	1022
Certificados do cliente	1029
AWS WAF	1070
Limitação	1073
APIs REST privadas	1076
Monitorar	1134
Métricas do CloudWatch	1135
Logs do CloudWatch	1145
Firehose	1150
Variáveis para registro em log de acesso no API Gateway	1152
X-Ray	1167
Portais do API Gateway	1181
Considerações	1181
Produto do portal	1182
Exemplo de produto do portal de adoção de animais de estimação	1182
Próximas etapas	1184
Criar um produto do portal	1185
Criar um endpoint REST de produto	1187
Criar uma página de produto	1189
Atualizar um produto do portal	1191
Atualizar um endpoint REST de produto	1191
Atualizar a página do produto	1192
Compartilhar produtos do portal	1193

Habilitar o botão Experimente para um endpoint REST do produto em seu portal	1199
Excluir um produto do portal	1200
Criar um portal	1201
Escolher um nome do domínio	1201
Criar um portal	1202
Atualizar um portal	1204
Atualizar um portal	1204
Visualizar um portal	1205
Considerações	1205
Visualizar um portal	1205
Publicar um portal	1206
Considerações	1206
Publicar um portal	1206
Usar um portal	1206
Usar um portal	1207
Solucionar problemas no portal	1207
Desabilitar um portal	1208
Desabilitar um portal	1208
Excluir um portal	1209
Excluir um portal	1209
APIs HTTP do API Gateway	1210
Desenvolver	1210
Criar uma API HTTP	1211
Rotas	1212
Tipos de endereço IP para APIs HTTP no API Gateway	1215
Controle de acesso	1217
Integrações	1236
CORS	1255
Mapeamento de parâmetros	1258
OpenAPI	1265
Publicar	1275
Stages	1276
Política de segurança para APIs HTTP no API Gateway	1278
Nomes de domínios personalizados	1280
Proteger	1290
Controle de utilização	1290

TLS mútuo	1291
Monitor	1298
Metrics	1299
Registro em log	1301
Solução de problemas	1312
Integrações do Lambda	1312
Autorizadores de JWT	1315
APIs de WebSocket do API Gateway	1317
Visão geral das APIs de WebSocket	1317
Usar rotas para uma API de WebSocket	1318
Enviar dados a aplicações de cliente conectadas	1319
Códigos de status API de WebSocket	1319
Gerenciar usuários conectados e aplicações clientes	1320
Invocar a integração do backend	1323
Expressões de seleção do WebSocket	1328
Desenvolver	1337
Criar e configurar	1337
Tipos de endereço IP para APIs de WebSocket no API Gateway	1339
Rotas	1341
Controle de acesso	1349
Integrações	1358
Validação da solicitação	1367
Transformações de dados	1371
Tipos de mídia binária	1384
Invocar	1384
Publicar	1388
Estágios	1388
Implantar uma API WebSocket	1391
Política de segurança para APIs de WebSocket no API Gateway	1394
Nomes de domínios personalizados	1396
Proteger	1403
Controle de utilização no nível da conta por região	1404
Limitação em nível de rota	1404
Monitor	1405
Métricas	1405
Registro em log	1407

ARNs do API Gateway	1416
Recursos de API HTTP e API WebSocket	1416
Recursos de API REST	1419
Recursos de nomes de domínio	1423
execute-api (APIs HTTP, WebSocket e REST)	1424
Extensões do OpenAPI	1426
x-amazon-apigateway-any-method	1427
Exemplos de x-amazon-apigateway-any-method	1428
x-amazon-apigateway-cors	1429
Exemplo de x-amazon-apigateway-cors	1429
x-amazon-apigateway-api-key-source	1430
Exemplo de x-amazon-apigateway-api-key-source	1431
x-amazon-apigateway-auth	1432
Exemplo de x-amazon-apigateway-auth	1432
x-amazon-apigateway-authorizer	1433
Exemplos de x-amazon-apigateway-authorizer para APIs REST	1436
Exemplos de x-amazon-apigateway-authorizer para APIs HTTP	1440
x-amazon-apigateway-authtype	1442
Exemplo de x-amazon-apigateway-authtype	1442
Consulte também	1445
x-amazon-apigateway-binary-media-type	1445
Exemplo de x-amazon-apigateway-binary-media-types	1445
x-amazon-apigateway-documentation	1445
Exemplo de x-amazon-apigateway-documentation	1445
x-amazon-apigateway-endpoint-access-mode	1446
x-amazon-apigateway-endpoint-access-modeExemplo de	1446
x-amazon-apigateway-endpoint-configuration	1446
Exemplos de x-amazon-apigateway-endpoint-configuration	1448
x-amazon-apigateway-gateway-responses	1448
Exemplo de x-amazon-apigateway-gateway-responses	1448
x-amazon-apigateway-gateway-responses.gatewayResponse	1449
Exemplo de x-amazon-apigateway-gateway-responses.gatewayResponse	1450
x-amazon-apigateway-gateway-responses.responseParameters	1450
x-amazon-apigateway-gateway-responses.responseParameters example	1451
x-amazon-apigateway-gateway-responses.responseTemplates	1451
Exemplo de x-amazon-apigateway-gateway-responses.responseTemplates	1452

x-amazon-apigateway-importexport-version	1452
x-amazon-apigateway-importexport-version example	1452
x-amazon-apigateway-integration	1453
Exemplos de x-amazon-apigateway-integration	1460
x-amazon-apigateway-integrations	1462
Exemplo de x-amazon-apigateway-integrations	1462
x-amazon-apigateway-integration.requestTemplates	1464
Exemplo de x-amazon-apigateway-integration.requestTemplates	1464
x-amazon-apigateway-integration.requestParameters	1465
x-amazon-apigateway-integration.requestParametersExemplo de	1466
x-amazon-apigateway-integration.responses	1467
x-amazon-apigateway-integration.responsesExemplo de	1468
x-amazon-apigateway-integration.response	1469
x-amazon-apigateway-integration.responseExemplo de	1470
x-amazon-apigateway-integration.responseTemplates	1471
Exemplo de x-amazon-apigateway-integration.responseTemplate	1471
x-amazon-apigateway-integration.responseParameters	1472
x-amazon-apigateway-integration.responseParametersExemplo de	1472
x-amazon-apigateway-integration.tlsConfig	1472
Exemplos x-amazon-apigateway-integration.tlsConfig	1474
x-amazon-apigateway-minimum-compression-size	1475
Exemplo de x-amazon-apigateway-minimum-compression-size	1475
x-amazon-apigateway-policy	1476
x-amazon-apigateway-policyExemplo de	1476
x-amazon-apigateway-request-validator	1477
x-amazon-apigateway-request-validatorExemplo de	1477
x-amazon-apigateway-request-validators	1478
x-amazon-apigateway-request-validatorsExemplo de	1478
x-amazon-apigateway-request-validators.requestValidator	1479
x-amazon-apigateway-request-validators.requestValidatorExemplo de	1479
x-amazon-apigateway-security-policy	1480
x-amazon-apigateway-security-policyExemplo de	1480
x-amazon-apigateway-tag-value	1480
x-amazon-apigateway-tag-valueExemplo de	1480
Segurança	1482
Proteção de dados	1483

Criptografia de dados	1484
Privacidade do tráfego entre redes	1486
Identity and Access Management	1487
Público	1487
Autenticação com identidades	1488
Gerenciar o acesso usando políticas	1489
Como o Amazon API Gateway funciona com o IAM	1491
Exemplos de políticas baseadas em identidade	1496
Exemplos de políticas baseadas em recursos	1508
Solução de problemas	1508
Uso de funções vinculadas a serviço	1510
Registro em log e monitoramento	1516
Trabalhar com o CloudTrail	1517
Trabalhar com o AWS Config	1521
Validação de compatibilidade	1524
Resiliência	1524
Segurança da infraestrutura	1525
Análise de configuração e vulnerabilidade	1526
Práticas recomendadas	1526
Atribuição de tags (tagging)	1528
Recursos do API Gateway que podem ser marcados	1529
Herança de tags na API do Amazon API Gateway V1	1530
Restrições de tags e convenções de uso	1531
Controle de acesso baseado em atributos	1532
Limitar ações com base em tags de recursos	1532
Permitir ações com base em tags de recursos	1533
Negar operações de marcação	1534
Permitir operações de marcação	1535
Referências de API	1537
Endpoints do serviço do API Gateway	1537
Endpoints IPv4	1537
Endpoints de pilha dupla (IPv4 e IPv6)	1538
Especificar endpoints	1538
Como usar endereços do IPv6 em políticas do IAM	1539
Cotas e observações importantes	1541
Cotas no nível da conta do API Gateway, por região	1541

Cotas do API Gateway para criação, implantação e gerenciamento de uma API	1542
Cotas de API REST	1546
Práticas recomendadas para reduzir cotas	1552
Cotas de API HTTP	1554
Práticas recomendadas para reduzir cotas	1557
Cotas de portal	1557
Cotas da API de WebSocket	1559
Observações importantes	1561
Observações importantes sobre o Amazon API Gateway para APIs HTTP	1562
Observações importantes do Amazon API Gateway para HTTP e APIs de WebSocket	1562
Observações importantes para APIs REST e APIs de WebSocket	1562
Observações importantes para APIs do WebSocket	1563
Observações importantes para APIs REST	1563
Histórico do documento	1570
Atualizações anteriores	1586

O que é o Amazon API Gateway?

O Amazon API Gateway é um serviço da AWS para criação, publicação, manutenção, monitoramento e proteção de APIs REST e WebSocket em qualquer escala. Os desenvolvedores de API podem criar APIs que acessem a AWS ou outros web services, bem como dados armazenados na [Nuvem AWS](#). Como um desenvolvedor de APIs do API Gateway, é possível criar APIs para uso em suas próprias aplicações cliente. Ou você pode disponibilizar suas APIs para desenvolvedores de aplicativos de terceiros. Para obter mais informações, consulte [the section called “Quem usa o API Gateway?”](#).

O API Gateway cria APIs RESTful que:

- São baseadas em HTTP.
- Habilitam a comunicação cliente-servidor sem estado.
- Implementam os métodos HTTP padrão, como GET, POST, PUT, PATCH e DELETE.

Para obter mais informações sobre APIs REST do API Gateway e APIs HTTP, consulte [the section called “Escolher entre APIs REST e APIs HTTP”](#), [APIs HTTP do API Gateway](#), [the section called “Usar o API Gateway para criar APIs REST”](#) e [the section called “Desenvolver”](#).

O API Gateway cria APIs WebSocket que:

- Seguem o protocolo [WebSocket](#), que permite a comunicação full-duplex entre cliente e servidor com estado.
- Roteiam mensagens recebidas com base no conteúdo da mensagem.

Para obter mais informações sobre APIs WebSocket do API Gateway, consulte [the section called “Usar o API Gateway para criar APIs do WebSocket”](#) e [the section called “Visão geral das APIs de WebSocket”](#).

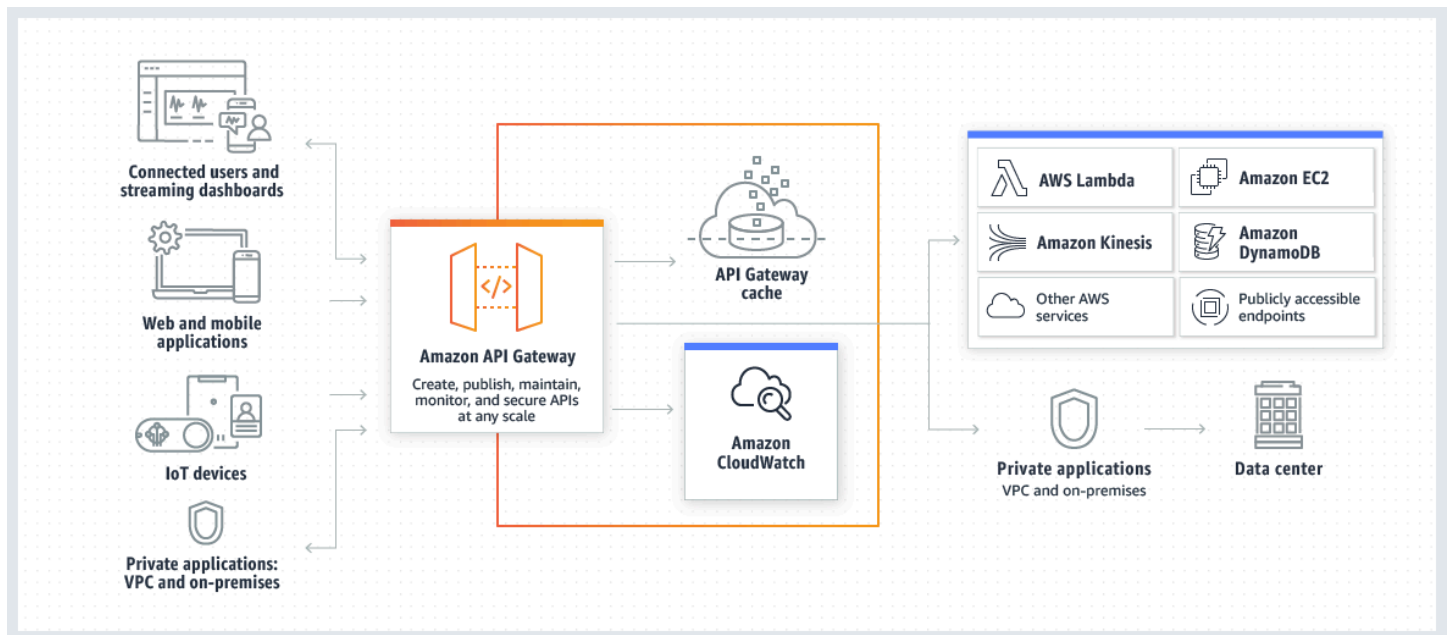
Tópicos

- [Arquitetura do API Gateway](#)
- [Recursos do API Gateway](#)
- [Casos de uso do API Gateway](#)
- [Acessar o API Gateway](#)

- [Parte da infraestrutura sem servidor da AWS](#)
- [Saiba como começar a usar o Amazon API Gateway](#)
- [Conceitos do Amazon API Gateway](#)
- [Escolher entre APIs REST e APIs HTTP](#)
- [Começar a usar o console da API REST](#)

Arquitetura do API Gateway

O diagrama a seguir mostra a arquitetura do API Gateway.



Este diagrama ilustra como as APIs criadas no Amazon API Gateway proporcionam a você ou a seus clientes desenvolvedores uma experiência de desenvolvedor integrada e consistente para criar aplicações sem servidor da AWS. O API Gateway processa todas as tarefas relacionadas à aceitação e ao processamento de até centenas de milhares de chamadas simultâneas de APIs. As tarefas incluem gerenciamento de tráfego, controle de autorização e acesso, monitoramento e gerenciamento de versão de APIs.

O API Gateway atua como uma “porta frontal” para aplicações acessarem dados, lógica de negócios ou funcionalidade de seus serviços de backend, como cargas de trabalho executadas no Amazon Elastic Compute Cloud (Amazon EC2), código executado no AWS Lambda, qualquer aplicação Web ou aplicações de comunicação em tempo real.

Recursos do API Gateway

O Amazon API Gateway oferece recursos como os seguintes:

- Suporte para APIs com estado ([WebSocket](#)) e sem estado ([HTTP](#)) e [REST](#).
- Mecanismos de [autenticação](#) poderosos e flexíveis, como políticas de , funções do autorizador do e grupos de usuários do AWS Identity and Access Management.
- [Implantações de versão Canary](#) para lançar alterações com segurança.
- Registro em log e monitoramento do [CloudTrail](#) do uso e alterações de API.
- Registro de acesso em logs e registro de execução em logs do CloudWatch, incluindo a capacidade de definir alarmes. Para obter mais informações, consulte [the section called “Métricas do CloudWatch”](#) e [the section called “Métricas”](#).
- Capacidade de usar modelos do CloudFormation para permitir a criação de APIs. Para obter mais informações, consulte [Referência de tipos de recursos do Amazon API Gateway](#) e [Referência de tipos de recursos do Amazon API Gateway V2](#).
- Suporte para [nomes de domínio personalizados](#).
- Integração ao [AWS WAF](#) para proteger suas APIs contra explorações comuns da web.
- Integração com latências de desempenho [AWS X-Ray](#) para compreensão e triagem.

Para obter uma lista completa das versões de recursos do API Gateway, consulte [Histórico do documento](#).

Casos de uso do API Gateway

A seção de casos de uso a seguir apresenta uma visão geral dos diferentes tipos de API do API Gateway e dos diferentes tipos de desenvolvedor que usam o API Gateway. Para ter mais informações sobre a diferença entre as APIs HTTP e as APIs REST, consulte [the section called “Escolher entre APIs REST e APIs HTTP”](#).

Tópicos

- [Usar o API Gateway para criar APIs REST](#)
- [Usar o API Gateway para criar APIs HTTP](#)
- [Usar o API Gateway para criar APIs do WebSocket](#)

- [Quem usa o API Gateway?](#)

Usar o API Gateway para criar APIs REST

Uma API REST do API Gateway é composta por recursos e métodos. Um recurso é uma entidade lógica que um aplicativo pode acessar através de um caminho de recurso. Um método corresponde a uma solicitação de API REST enviada pelo usuário de sua API e a resposta retornada ao usuário.

Por exemplo, `/incomes` pode ser o caminho de um recurso que representa a renda do usuário do aplicativo. Um recurso pode ter uma ou mais operações que são definidas por verbos HTTP adequados, como GET, POST, PUT, PATCH e DELETE. Uma combinação de um caminho de recurso e uma operação identifica um método da API. Por exemplo, um método POST `/incomes` pode adicionar uma renda obtida pelo autor da chamada, e um método GET `/expenses` pode consultar as despesas relatadas incorridas pelo autor da chamada.

O aplicativo não precisa saber onde os dados solicitados estão armazenados e de onde são obtidos no backend. Nas APIs REST do API Gateway, o front-end é encapsulado por solicitações de método e respostas do método. A API faz interface com o backend usando as solicitações de integração e respostas de integração.

Por exemplo, com o DynamoDB como o backend, o desenvolvedor de APIs configura a solicitação de integração para encaminhar a solicitação do método de entrada para o backend escolhido. A configuração inclui especificações de uma ação adequada do DynamoDB, a função e as políticas do IAM necessárias e os dados de entrada necessários. O backend retorna o resultado para o API Gateway como uma resposta da integração.

Para rotear a resposta da integração para uma resposta de método adequada (de um determinado código de status HTTP) para o cliente, você pode configurar a resposta da integração para mapear os parâmetros de resposta necessários da integração para o método. Em seguida, você converte o formato dos dados de saída do backend para o formato do front-end, se necessário. O API Gateway permite que você defina um esquema ou um modelo para a [carga](#) a fim de facilitar a configuração do modelo de mapeamento do corpo.

O API Gateway fornece a funcionalidade de gerenciamento da API REST da seguinte forma:

- Suporte para gerar SDKs e criar a documentação da API usando extensões do API Gateway para OpenAPI
- Limitação de solicitações HTTP

Usar o API Gateway para criar APIs HTTP

As APIs HTTP permitem criar APIs RESTful com menor latência e menor custo que as APIs REST.

Você pode usar APIs HTTP para enviar solicitações para funções do AWS Lambda ou para qualquer endpoint HTTP roteável.

Por exemplo, você pode criar uma API HTTP que se integre a uma função do Lambda no backend. Quando um cliente chama sua API, o API Gateway envia a solicitação para a função do Lambda e retorna a resposta da função para o cliente.

APIs HTTP são compatíveis com as autorizações [OpenID Connect](#) e [OAuth 2.0](#). Eles são fornecidos com suporte incorporado para compartilhamento de recursos de origem cruzada (CORS) e implantações automáticas.

Para saber mais, consulte [the section called “Escolher entre APIs REST e APIs HTTP”](#).

Usar o API Gateway para criar APIs do WebSocket

Em uma API WebSocket, o cliente e o servidor podem trocar mensagens entre si a qualquer momento. Os servidores de backend podem enviar dados para usuários e dispositivos conectados com facilidade, o que evita a necessidade de implementar mecanismos complexos de sondagem.

Por exemplo, é possível criar uma aplicação sem servidor usando uma API WebSocket do API Gateway e do AWS Lambda para enviar e receber mensagens de e para usuários individuais ou grupos de usuários em uma sala de conversa. Você também pode invocar serviços de backend como o AWS Lambda, o Amazon Kinesis ou um endpoint HTTP com base no conteúdo da mensagem.

É possível usar as APIs WebSocket do API Gateway para criar aplicações de comunicação segura e em tempo real sem a necessidade de provisionar ou gerenciar os servidores para administrar conexões ou intercâmbios de dados em grande escala. Casos de uso direcionados incluem aplicativos em tempo real, como:

- Aplicativos de bate-papo
- Painéis em tempo real, como cotações de ações
- Alertas e notificações em tempo real

O API Gateway fornece a funcionalidade de gerenciamento de API WebSocket, como:

- Monitoramento e limitações de conexões e mensagens
- Uso do AWS X-Ray para rastrear mensagens à medida que são transferidas por meio de APIs para serviços de backend
- Fácil integração com endpoints HTTP/HTTPS

Quem usa o API Gateway?

Há dois tipos de desenvolvedores que usam o API Gateway: desenvolvedores de APIs e desenvolvedores de aplicações.

Um desenvolvedor de APIs cria e implanta uma API para habilitar a funcionalidade necessária no API Gateway. O desenvolvedor de APIs deve ser um usuário na conta da AWS que é a proprietária da API.

Um desenvolvedor de aplicações cria uma aplicação funcional para chamar serviços da AWS ao invocar uma API WebSocket ou REST criada por um desenvolvedor de APIs no API Gateway.

O desenvolvedor de aplicativos é o cliente do desenvolvedor de APIs. O desenvolvedor de aplicações não precisa ter uma conta da AWS, desde que a API não exija permissões do IAM nem ofereça suporte à autorização de usuários por meio de provedores de identidade federada de terceiros compatíveis com a [Federação de identidades do grupo de usuários do Amazon Cognito](#). Esses provedores de identidade incluem a Amazon, o grupo de usuários do Amazon Cognito, o Facebook e o Google.

Criar e gerenciar uma API do API Gateway

Um desenvolvedor de APIs trabalha com o componente de serviço do API Gateway para gerenciamento de APIs, denominado `apigateway`, para criar, configurar e implantar uma API.

Como desenvolvedor de APIs, é possível criar e gerenciar uma API usando o console do API Gateway, descrito em [Começar a usar o API Gateway](#) ou chamando [Referências de API](#). Há várias maneiras de chamar essa API. Elas incluem o uso da AWS Command Line Interface (AWS CLI) ou de um AWS SDK. Além disso, você pode habilitar a criação de API com [modelos AWS CloudFormation](#) ou (no caso de APIs REST e APIs HTTP) [Extensões OpenAPI para o API Gateway](#).

Para obter uma lista de regiões em que o API Gateway está disponível, bem como os endpoints de serviço de controle associados, consulte [Endpoints e cotas do Amazon API Gateway](#).

Chamar uma API do API Gateway

Um desenvolvedor de aplicativos trabalha com o componente de serviço do API Gateway para execução da API, denominado `execute-api`, para invocar uma API criada ou implantada no API Gateway. As entidades de programação subjacentes são expostas pela API criada. Há várias maneiras de chamar esse tipo de API. Para saber mais, consulte [Invocar APIs REST no API Gateway](#) e [Invocar APIs de WebSocket](#).

Acessar o API Gateway

É possível acessar o Amazon API Gateway das seguintes maneiras:

- Console de gerenciamento da AWS: o Console de gerenciamento da AWS fornece uma interface da Web para criar e gerenciar APIs. Após concluir as etapas em [Pré-requisitos](#), você pode acessar o console do API Gateway em <https://console.aws.amazon.com/apigateway>.
- AWS SDKs: se você estiver usando uma linguagem de programação para a qual a AWS fornece um SDK, você poderá usar um SDK para acessar o API Gateway. Os SDKs simplificam a autenticação, integram-se com facilidade ao ambiente de desenvolvimento e fornecem acesso aos comandos do API Gateway. Para mais informações, consulte [Ferramentas para a Amazon Web Services](#).
- APIs do API Gateway V1 e V2: se você estiver usando uma linguagem de programação para a qual não haja SDK disponível, consulte a [Referência de API do Amazon API Gateway versão 1](#) e [Referência de API do Amazon API Gateway versão 2](#).
- AWS Command Line Interface: para obter mais informações, consulte [Noções básicas de configuração do AWS Command Line Interface](#) no Guia do usuário do AWS Command Line Interface.
- AWS Tools for Windows PowerShell: para obter mais informações, consulte [Noções básicas de configuração do AWS Tools for Windows PowerShell](#) no Guia do usuário do Ferramentas da AWS para PowerShell.

Parte da infraestrutura sem servidor da AWS

Juntamente com o [AWS Lambda](#), o API Gateway forma a parte voltada para a aplicação da infraestrutura sem servidor da AWS. Para saber mais sobre como começar a usar a tecnologia sem servidor, consulte o [Guia do desenvolvedor da tecnologia sem servidor](#).

Para que uma aplicação chame serviços da AWS publicamente disponíveis, você pode usar o Lambda para interagir com os serviços necessários e expor as funções do Lambda por meio de métodos da API no API Gateway. O AWS Lambda executa seu código em uma infraestrutura de computação altamente disponível. Ele realiza a execução e a administração necessárias dos recursos de computação. Para habilitar aplicações sem servidor, o API Gateway oferece [suporte a integrações de proxy simplificadas](#) com AWS Lambda e endpoints HTTP.

Saiba como começar a usar o Amazon API Gateway

Para obter uma introdução ao Amazon API Gateway, consulte o seguinte:

- [Conceitos básicos](#), que fornece uma demonstração para a criação de uma API HTTP.
- [Iaas sem servidor](#), que fornece vídeos instrucionais.
- [Happy Little API Shorts](#), que é uma série de breves vídeos instrutivos.

Conceitos do Amazon API Gateway

A seção a seguir descreve conceitos introdutórios para o uso do API Gateway.

API Gateway

O API Gateway é um serviço da AWS que oferece suporte à:

- Criação, implantação e gerenciamento de uma interface de programação de aplicativos (API) [RESTful](#) para expor endpoints HTTP de backend, funções do AWS Lambda ou outros serviços da AWS.
- Criação, implantação e gerenciamento de uma API [WebSocket](#) para expor funções do AWS Lambda ou outros serviços da AWS.
- Invoque métodos de API expostos por meio de endpoints HTTP e WebSocket de front-end.

API REST do API Gateway

Uma coleção de métodos e recursos HTTP que são integrados a endpoints HTTP de backend, funções do Lambda ou outros serviços da AWS. É possível implantar essa coleção em um ou mais estágios. Normalmente, os recursos da API são organizados em uma árvore de recursos, de acordo com a lógica do aplicativo. Cada recurso de API pode expor um ou mais métodos de API, que têm verbos HTTP exclusivos compatíveis com o API Gateway. Para obter mais informações, consulte [the section called “Escolher entre APIs REST e APIs HTTP”](#).

API HTTP do API Gateway

Uma coleção de rotas e métodos que são integrados a endpoints HTTP de backend ou funções do Lambda. É possível implantar essa coleção em um ou mais estágios. Cada rota pode expor um ou mais métodos de API, que têm verbos HTTP exclusivos compatíveis com o API Gateway. Para obter mais informações, consulte [the section called “Escolher entre APIs REST e APIs HTTP”](#).

API WebSocket do API Gateway

Uma coleção de rotas WebSocket e de chaves de rota que são integradas a endpoints HTTP de backend, funções do Lambda ou outros serviços da AWS. É possível implantar essa coleção em um ou mais estágios. Os métodos de API são invocados por meio de conexões de endpoints WebSocket de front-end que você pode associar a um nome de domínio personalizado registrado.

Implantação de API

Um snapshot point-in-time da sua API do API Gateway. Para disponibilizar para os clientes, a implantação deve ser associada a um ou mais estágios de API.

Desenvolvedor de API

Sua conta da AWS que é dona de uma implantação do API Gateway (por exemplo, um provedor de serviços que também oferece suporte a acesso por programação.)

Endpoint de API

Um nome de host para uma API no API Gateway que é implantada em uma região específica. O nome de host tem a seguinte forma `{api-id}.execute-api.{region}.amazonaws.com`. Há suporte para os seguintes tipos de endpoints de API:

- [Endpoint de API otimizada para fronteiras](#)
- [Endpoint privado de API](#)
- [Endpoint de API regional](#)

Chave da API

Uma string alfanumérica usada pelo API Gateway para identificar um desenvolvedor de aplicações que usa sua API REST ou WebSocket. O API Gateway pode gerar chaves de API em seu nome ou você pode importá-las para um arquivo CSV. É possível usar chaves de API com [autorizadores do Lambda](#) ou [planos de uso](#) para controlar o acesso às suas APIs.

Consulte [Endpoints de API](#).

Proprietário de API

Consulte [Desenvolvedor de APIs](#).

Estágio de API

Referência lógica a um estado do ciclo de vida de sua API (por exemplo, 'dev', 'prod', 'beta', 'v2'). Os estágios de API são identificados pelo ID da API e pelo nome do estágio.

Desenvolvedor de aplicativos

Um criador de aplicativos que pode ou não ter uma conta da AWS e interage com a API que você, o desenvolvedor da API, implantou. Os desenvolvedores de aplicativos são seus clientes. Um desenvolvedor de aplicativos normalmente é identificado por uma [chave da API](#).

URL de retorno de chamada

Quando um novo cliente é conectado por meio de uma conexão WebSocket, você pode chamar uma integração no API Gateway para armazenar o URL de retorno de chamada do cliente. Você poderá então usar este URL de retorno de chamada para enviar mensagens para o cliente a partir do sistema de backend.

Portal do desenvolvedor

Uma aplicação em que os provedores de API podem compartilhar suas APIs e a respectiva documentação com os consumidores da API. As APIs são agrupadas em produtos, que são uma coleção de endpoints da API REST, documentação da API e documentação complementar do produto.

Consulte [Portais do API Gateway](#).

Endpoint de API otimizada para fronteiras

O nome do host padrão de uma API do API Gateway implantada na região especificada usando uma distribuição do CloudFront para facilitar o acesso de clientes normalmente entre regiões da AWS. As solicitações de API são roteadas para o ponto de presença (POP) do CloudFront mais próximo, o que geralmente melhora o tempo de conexão para clientes em regiões diferentes.

Consulte [Endpoints de API](#).

Solicitação de integração

A interface interna de uma rota de API WebSocket ou de um método de API REST no API Gateway, na qual você mapeia o corpo de uma solicitação de rota ou os parâmetros e o corpo de uma solicitação de método para os formatos exigidos pelo backend.

Resposta de integração

A interface interna de um rota de API WebSocket ou de um método de API REST no API Gateway na qual você mapeia os códigos de status, os cabeçalhos e a carga que são recebidos do backend para o formato de resposta que é retornado a um aplicativo cliente.

modelo de mapeamento

Um script em [Velocity Template Language \(VTL\)](#) que transforma um corpo de solicitação do formato de dados de front-end para o formato de dados de backend, ou que transforma um corpo de resposta do formato de dados de backend para o formato de dados de front-end. Os modelos de mapeamento podem ser especificados na solicitação de integração ou na resposta de integração. Eles podem fazer referência a dados disponibilizados em tempo de execução como contexto e variáveis de estágio.

O mapeamento pode ser tão simples quanto uma [transformação de identidade](#) que passa os cabeçalhos ou o corpo pela integração no estado em que se encontra do cliente para o backend para uma solicitação. O mesmo aplica-se para uma resposta, na qual a carga é passada do backend ao cliente.

Solicitação de método

A interface pública de um método de API no API Gateway que define os parâmetros e o corpo que um desenvolvedor de aplicativos deve enviar nas solicitações para acessar o backend por meio da API.

Resposta do método

A interface pública de uma API REST que define os códigos de status, os cabeçalhos e os modelos de corpo que um desenvolvedor de aplicativos deve esperar em respostas da API.

Integração simulada

Em uma integração simulada, respostas de API são geradas no API Gateway diretamente, sem a necessidade de um backend de integração. Como desenvolvedor de APIs, você decide como o API Gateway responde a uma solicitação de integração simulada. Para isso, você configura a solicitação de integração e a resposta de integração do método para associar uma resposta a um determinado código de status.

Modelo

Um esquema de dados especificando a estrutura de dados de uma solicitação ou carga de resposta. É necessário um modelo para gerar um SDK fortemente tipado de uma API. Ele

também é usado para validar as cargas. Um modelo é conveniente para gerar um modelo de mapeamento de amostra para iniciar a criação de um modelo de mapeamento de produção. Embora seja útil, um modelo não é necessário para criar um modelo de mapeamento.

Portal

Consulte [Portal do desenvolvedor](#).

Produto do portal

Serviço ou funcionalidade que você deseja compartilhar. Seu produto de portal consiste em endpoints REST e páginas do produto. Os endpoints REST do produto são os pontos de acesso ao seu produto do portal e consistem no caminho e no método de uma API REST e no estágio em que ela é implantada. As páginas do produto são a documentação que você fornece para explicar como os consumidores da API podem usar os endpoints do seu produto. Você pode compartilhar produtos entre contas da AWS para adicioná-los aos portais.

Consulte [Criar um produto](#).

API privado

Consulte [Endpoint privado de API](#).

Endpoint privado de API

Um endpoint de API que é exposto por meio de VPC endpoints de interface e permite que um cliente acesse de forma segura os recursos da API privada dentro de uma VPC. As APIs privadas são isoladas da Internet pública e só podem ser acessadas usando VPC endpoints para o API Gateway para o qual foi concedido acesso.

Integração privada

Um tipo de integração do API Gateway para um cliente acessar recursos dentro da VPC do cliente por meio de um endpoint de API REST privada sem expor os recursos à Internet pública.

Integração de proxy

Uma configuração simplificada de integração do API Gateway. É possível configurar uma integração de proxy como um tipo de integração de proxy HTTP ou uma integração de proxy do Lambda.

Para a integração de proxy HTTP, o API Gateway transmite toda a solicitação e a resposta entre o front-end e um backend HTTP. Para integração de proxy do Lambda, o API Gateway envia toda

a solicitação como entrada para uma função do Lambda de backend. Depois, o API Gateway transforma a saída da função do Lambda em uma resposta HTTP de front-end.

Nas APIs REST, a integração de proxy é mais comumente usada com um recurso de proxy, representado por uma variável de caminho voraz (por exemplo, `{proxy+}`), combinada com um método genérico ANY.

Criação rápida

É possível usar a criação rápida para simplificar a criação de uma API HTTP. A criação rápida cria uma API com uma integração ao Lambda ou a HTTP, uma rota genérica padrão e um estágio padrão configurado para implantar alterações automaticamente. Para obter mais informações, consulte [the section called “Criar uma API HTTP usando a CLI da AWS”](#).

Endpoint de API regional

O nome do host de uma API implantada na região especificada e com o objetivo de atender a clientes, como instâncias do EC2, na mesma região da AWS. As solicitações de API são destinadas diretamente para a API do API Gateway específico da região sem passar por nenhuma distribuição do CloudFront. Para solicitações em uma mesma região, um endpoint regional ignora a ida e volta desnecessárias para uma distribuição do CloudFront.

Além disso, você pode aplicar o [roteamento baseado em latência](#) em endpoints regionais para implantar uma API em várias regiões usando a mesma configuração de endpoint da API regional, definir o mesmo nome de domínio personalizado para cada API implantada e configurar registros DNS baseados em latência no Route 53 a fim de rotear solicitações do cliente para uma região cuja latência é a mais baixa.

Consulte [Endpoints de API](#).

Rota

Uma rota WebSocket no API Gateway é usada para direcionar mensagens recebidas para uma integração específica, como uma função do AWS Lambda com base no conteúdo da mensagem. Ao definir sua API WebSocket, você especifica uma chave de roteamento e um backend de integração. A chave de roteamento é um atributo no corpo da mensagem. Quando é feita a correspondência da chave de roteamento em uma mensagem recebida, o backend de integração é invocado.

Uma rota padrão também podem ser definida para chaves de roteamento que não façam correspondência ou para especificar um modelo de proxy que passa a mensagem no estado em

que se encontra para os componentes de backend que executam o roteamento e processam a solicitação.

Rotear solicitação

A interface pública de um método de API WebSocket no API Gateway que define o corpo que um desenvolvedor de aplicativos deve enviar nas solicitações para acessar o backend por meio da API.

Rotear resposta

A interface pública de uma API WebSocket que define os códigos de status, os cabeçalhos e os modelos de corpo que um desenvolvedor de aplicativos deve esperar do API Gateway.

Plano de uso

Um [plano de uso](#) fornece aos clientes da API selecionada acesso a uma ou mais APIs REST ou WebSocket implantadas. Você pode usar um plano de uso para configurar o controle e os limites de cota individuais que são aplicados em chaves de API individuais de cliente.

Conexão WebSocket

O API Gateway mantém uma conexão persistente entre os clientes e o próprio API Gateway. Não há conexão persistente entre o API Gateway e integrações de backend, como funções do Lambda. Os serviços de backend são invocados conforme necessário, com base no conteúdo das mensagens recebidas dos clientes.

Escolher entre APIs REST e APIs HTTP

APIs REST e APIs HTTP são produtos da API RESTful. As APIs REST são compatíveis com mais recursos do que as APIs HTTP, enquanto as APIs HTTP são projetadas com recursos mínimos para que possam ser oferecidas por um preço mais baixo. Escolha APIs REST se precisar de recursos como chaves de API, limitação por cliente, validação de solicitações, integração AWS WAF ou endpoints de API privados. Escolha APIs HTTP se você não precisar dos recursos incluídos nas APIs REST.

As seções a seguir resumem os principais recursos que estão disponíveis em APIs HTTP e REST. Quando necessário, links adicionais são fornecidos para navegar entre as seções da API REST e da API HTTP do Guia do desenvolvedor do API Gateway.

Tipo de endpoint

O tipo de endpoint refere-se ao endpoint que o API Gateway cria para sua API. Para obter mais informações, consulte [the section called “Tipos de endpoint do API Gateway”](#).

Tipos de endpoint	API REST	API HTTP
Otimizado para borda	Sim	No (Não)
Regional	Yes (Sim)	Yes (Sim)
Privado	Sim	No (Não)

Segurança

O API Gateway fornece várias maneiras de proteger sua API de determinadas ameaças, como usuários mal-intencionados ou picos de tráfego. Para saber mais, consulte [the section called “Proteger”](#) e [the section called “Proteger”](#).

Recursos de segurança	API REST	API HTTP
Autenticação TLS mútua	Yes (Sim)	Yes (Sim)
Certificados para autenticação de back-end	Sim	Não
AWS WAF	Sim	No (Não)

Autorização

O API Gateway oferece suporte a vários mecanismos de controle e gerenciamento de acesso à sua API. Para obter mais informações, consulte [the section called “Controle de acesso”](#) e [the section called “Controle de acesso”](#).

Opções de autorização	API REST	API HTTP
IAM	Yes (Sim)	Yes (Sim)

Opções de autorização	API REST	API HTTP
Políticas de recursos	Sim	No (Não)
Amazon Cognito	Sim	Sim ¹
Autorização personalizada com uma função do AWS Lambda	Yes (Sim)	Yes (Sim)
JSON Web Token (JWT) ²	Não	Sim

¹ É possível usar o Amazon Cognito com um [Autorizador do JWT](#).

² É possível usar um [autorizador do Lambda](#) para validar JWTs para APIs REST.

Gerenciamento de APIs

Escolha APIs REST se precisar de recursos de gerenciamento de API, como chaves de API e limitação de taxa por cliente. Para obter mais informações, consulte [the section called “Distribuir”](#), [the section called “Nomes de domínios personalizados”](#) e [the section called “Nomes de domínios personalizados”](#).

Recursos	API REST	API HTTP
Domínios personalizados	Yes (Sim)	Yes (Sim)
Chaves de API	Sim	No (Não)
Limitação de taxa por cliente	Sim	No (Não)
Limitação de uso por cliente	Sim	No (Não)
Portal do desenvolvedor	Sim	No (Não)

Desenvolvimento

Ao desenvolver a API do API Gateway, você decidirá uma série de características da API. Essas características dependem do caso de uso da sua API. Para obter mais informações, consulte [the section called “Desenvolver”](#) e [the section called “Desenvolver”](#).

Recursos	API REST	API HTTP
Configuração de CORS	Yes (Sim)	Yes (Sim)
Testar invocações	Sim	No (Não)
Armazenamento em cache	Sim	No (Não)
Implantações controladas pelo usuário	Yes (Sim)	Yes (Sim)
Implantações automáticas	Não	Sim
Custom gateway responses (Respostas personalizadas do gateway)	Sim	No (Não)
Implantações da versão canário	Sim	No (Não)
Validação da solicitação	Sim	No (Não)
Solicitar transformação de parâmetros	Yes (Sim)	Yes (Sim)
Solicitar transformação do corpo	Sim	No (Não)

Monitoramento

O API Gateway é compatível com várias opções para registrar solicitações de API e monitorar suas APIs. Para obter mais informações, consulte [the section called “Monitorar”](#) e [the section called “Monitor”](#).

Recurso	API REST	API HTTP
Métricas do Amazon CloudWatch	Yes (Sim)	Yes (Sim)
Logs de acesso ao CloudWatch Logs	Yes (Sim)	Yes (Sim)
Logs de acesso ao Amazon Data Firehose	Sim	No (Não)
Logs de execução	Sim	No (Não)
AWS X-Ray Rastreamento do	Sim	No (Não)

Integrações

As integrações conectam sua API do API Gateway aos recursos de back-end. Para obter mais informações, consulte [the section called “Integrações”](#) e [the section called “Integrações”](#).

Recurso	API REST	API HTTP
Endpoints HTTP públicos	Yes (Sim)	Yes (Sim)
AWS Serviços da	Yes (Sim)	Yes (Sim)
AWS Lambda Funções do	Yes (Sim)	Yes (Sim)
Integrações privadas com Network Load Balancers	Yes (Sim)	Yes (Sim)
Integrações privadas com Application Load Balancers	Yes (Sim)	Yes (Sim)
Integrações privadas com AWS Cloud Map	Não	Sim
Integrações simuladas	Sim	No (Não)

Recurso	API REST	API HTTP
Streaming de respostas	Sim	No (Não)

Começar a usar o console da API REST

Neste exercício de conceitos básicos, você cria uma API REST sem servidor usando o console da API REST do API Gateway. As APIs sem servidor permitem que você se concentre nas aplicações, em vez de gastar tempo provisionando e gerenciando servidores. Este exercício leva menos de 20 minutos para ser concluído e é possível dentro do [nível gratuito da AWS](#).

Primeiro, crie uma função do Lambda usando o console do Lambda. Depois, crie uma API REST usando o console da API REST do API Gateway. Depois, crie um método de API e o integre a uma função do Lambda usando uma integração de proxy do Lambda. Por fim, implante e invoque a API.

Quando você invoca a API REST, o API Gateway encaminha a solicitação à função do Lambda. O Lambda executa a função e retorna uma resposta ao API Gateway. Então, o API Gateway retorna uma resposta para você.



Para concluir este exercício, você precisa de uma Conta da AWS e de um usuário do AWS Identity and Access Management (IAM) com acesso ao console. Para obter mais informações, consulte [Configurar para usar o API Gateway](#).

Tópicos

- [Etapa 1: Criar uma função do Lambda](#)
- [Etapa 2: criar uma API REST](#)
- [Etapa 3: criar uma integração de proxy do Lambda](#)
- [Etapa 4: implantar a API](#)
- [Etapa 5: invocar a API](#)
- [Etapa 6 \(opcional\): limpar](#)

Etapa 1: Criar uma função do Lambda

Você usa uma função do Lambda para o backend da sua API. O Lambda executa o código somente quando necessário e dimensiona automaticamente, desde algumas solicitações por dia a milhares por segundo.

Para este exercício, você usa a função Node.js padrão no console do Lambda.

Como criar uma função do Lambda

1. Abra o console do Lambda em <https://console.aws.amazon.com/lambda>
2. Escolha Create function (Criar função).
3. Em Basic information (Informações básicas), em Function name (Nome da função), insira **my-function**.
4. Em todas as outras opções, use a configuração padrão.
5. Selecione Criar função.

O código de função padrão do Lambda deve ser semelhante ao seguinte:

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

É possível modificar a função do Lambda para este exercício, desde que a resposta da função esteja alinhada com o [formato que o API Gateway requer](#).

Substitua o corpo da resposta padrão (Hello from Lambda!) por The API Gateway REST API console is great!. Quando você invoca a função de exemplo, ela retorna uma resposta 200 aos clientes, bem como a resposta atualizada.

Etapa 2: criar uma API REST

Em seguida, crie uma API REST com um recurso raiz (/).

Como criar uma API REST

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Execute um destes procedimentos:
 - Para criar sua primeira API, em API REST, escolha Criar.
 - Se você criou uma API antes, escolha Criar API e, depois, escolha Criar para API REST.
3. Em API name (Nome da API), insira **my-rest-api**.
4. (Opcional) Em Description (Descrição), insira uma descrição.
5. Mantenha Tipo de endpoint de API definido como Regional.
6. Em Tipo de endereço IP, selecione IPv4.
7. Selecione Criar API.

Etapa 3: criar uma integração de proxy do Lambda

Em seguida, você cria um método de API para a API REST no recurso raiz (/) e integra o método à função do Lambda usando uma integração de proxy. Em uma integração de proxy do Lambda, o API Gateway passa a solicitação de entrada do cliente diretamente para a função do Lambda.

Como criar uma integração de proxy do Lambda

1. Selecione o recurso / e escolha Criar método.
2. Em Tipo de método, selecione ANY.
3. Em Tipo de integração, selecione Lambda.
4. Ative Integração de proxy do Lambda.
5. Em Função do Lambda, insira **my-function** e selecione sua função do Lambda.
6. Escolha Criar método.

Etapa 4: implantar a API

Em seguida, você cria uma implantação de API e a associa a um estágio.

Para implantar sua API

1. Escolha Implantar API.
2. Em Estágio, selecione Novo estágio.

3. Em Stage name (Nome do estágio), insira **Prod**.
4. (Opcional) Em Description (Descrição), insira uma descrição.
5. Escolha Implantar.

Agora, os clientes podem chamar sua API. Para testar a API antes de implantá-la, você pode, opcionalmente, escolher o método QUALQUER, navegar até a guia Testar e escolher Testar.

Etapa 5: invocar a API

Como invocar a API

1. No painel de navegação principal, escolha Estágio.
2. Em Detalhes do estágio, escolha o ícone de cópia para copiar o URL de invocação da API.

Stage details [Info](#) Edit

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

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

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

3. Insira o URL de invocação em um navegador da web.

O URL deve ser semelhante a `https://abcd123.execute-api.us-east-2.amazonaws.com/Prod`.

Seu navegador envia uma solicitação GET à API.

4. Verifique a resposta da sua API. Você deve ver o texto "The API Gateway REST API console is great!" no seu navegador.

Etapa 6 (opcional): limpar

Para evitar acumular custos desnecessários para sua Conta da AWS, exclua os recursos que você criou como parte deste exercício. As etapas a seguir excluem a API REST, a função do Lambda e recursos associados.

Como excluir a API REST

1. No painel Recursos, escolha Ações de API, Excluir API.
2. Na caixa de diálogo Excluir API, insira confirmar e Excluir.

Como excluir uma função do Lambda

1. Abra o console do Lambda em <https://console.aws.amazon.com/lambda>
2. Na página Funções, selecione a função. Selecione Ações, Excluir.
3. Na caixa de diálogo Excluir 1 funções, insira **delete** e selecione Excluir.

Como excluir um grupo de logs da função do Lambda

1. Abra a [página Log groups](#) (Grupos de log) do console do Amazon CloudWatch.
2. Na página Grupos de logs, selecione o grupo de logs da função (/aws/lambda/my-function). Em Ações, escolha Excluir grupo(s) de logs.
3. Na caixa de diálogo Delete log group(s) (Excluir grupo(s) de logs), escolha Delete (Excluir).

Para excluir o perfil de execução de uma função do Lambda

1. Abra a página [Funções](#) no console do IAM.
2. (Opcional) Na página Perfis, na caixa de pesquisa, insira **my-function**.
3. Selecione o perfil da função (por exemplo, my-function-*31exxmpl*) e escolha Excluir.
4. Na caixa de diálogo Excluir **my-function-31exxmpl?**, insira o nome do perfil e selecione Excluir.

 Tip

Você pode automatizar a criação e a limpeza dos recursos da AWS usando o AWS CloudFormation ou o AWS Serverless Application Model (AWS SAM). Para ver alguns exemplos de modelos do CloudFormation, consulte os [exemplos de modelo para o API Gateway](#) no repositório awsdocs do GitHub.

Configurar para usar o API Gateway

Antes de usar o Amazon API Gateway pela primeira vez, execute as tarefas a seguir.

Inscrever-se para uma Conta da AWS

Se você ainda não tem uma Conta da AWS, siga as etapas abaixo para criar uma.

Como cadastrar uma Conta da AWS

1. Abra <https://portal.aws.amazon.com/billing/signup>.
2. Siga as instruções online.

Parte do procedimento de inscrição envolve receber uma chamada telefônica ou uma mensagem de texto e inserir um código de verificação pelo teclado do telefone.

Quando você se inscreve para uma Conta da AWS, um Usuário raiz da conta da AWS é criado. O usuário-raiz tem acesso a todos os Serviços da AWS e recursos na conta. Como prática recomendada de segurança, atribua o acesso administrativo a um usuário e use somente o usuário-raiz para executar [tarefas que exigem acesso de usuário-raiz](#).

A AWS envia um e-mail de confirmação depois que o processo de inscrição é concluído. A qualquer momento, é possível exibir as atividades da conta atual e gerenciar sua conta acessando <https://aws.amazon.com/> e selecionando Minha conta.

Criar um usuário com acesso administrativo

Depois de se cadastrar em uma Conta da AWS, proteja seu Usuário raiz da conta da AWS, habilite o Centro de Identidade do AWS IAM e crie um usuário administrativo para não usar o usuário-raiz em tarefas cotidianas.

Proteger o Usuário raiz da conta da AWS

1. Faça login no [Console de gerenciamento da AWS](#) como o proprietário da conta ao escolher a opção Usuário-raiz e inserir o endereço de e-mail da Conta da AWS. Na próxima página, insira a senha.

Para obter ajuda ao fazer login usando o usuário-raiz, consulte [Fazer login como usuário-raiz](#) no Guia do usuário do Início de Sessão da AWS.

2. Habilite a autenticação multifator (MFA) para o usuário-raiz.

Para obter instruções, consulte [Habilitar um dispositivo MFA virtual para sua Conta da AWS de usuário-raiz \(console\)](#) no Guia do usuário do IAM.

Criar um usuário com acesso administrativo

1. Habilita o Centro de Identidade do IAM.

Para obter instruções, consulte [Habilitar o Centro de Identidade do AWS IAM](#) no Guia do usuário do Centro de Identidade do AWS IAM.

2. No Centro de Identidade do IAM, conceda o acesso administrativo a um usuário.

Para obter um tutorial sobre como usar o Diretório do Centro de Identidade do IAM como a fonte de identidade, consulte [Configurar o acesso dos usuários com o Diretório do Centro de Identidade do IAM padrão](#) no Guia do usuário do Centro de Identidade do AWS IAM.

Iniciar sessão como o usuário com acesso administrativo

- Para fazer login com o seu usuário do Centro de Identidade do IAM, use o URL de login enviado ao seu endereço de e-mail quando o usuário do Centro de Identidade do IAM foi criado.

Para obter ajuda para fazer login usando um usuário do Centro de Identidade do IAM, consulte [Fazer login no portal de acesso da AWS](#), no Guia do usuário do Início de Sessão da AWS.

Atribuir acesso a usuários adicionais

1. No Centro de Identidade do IAM, crie um conjunto de permissões que siga as práticas recomendadas de aplicação de permissões com privilégio mínimo.

Para obter instruções, consulte [Criar um conjunto de permissões](#) no Guia do usuário do Centro de Identidade do AWS IAM.

2. Atribua usuários a um grupo e, em seguida, atribua o acesso de autenticação única ao grupo.

Para obter instruções, consulte [Adicionar grupos](#) no Guia do usuário do Centro de Identidade do AWS IAM.

Começar a usar o API Gateway

Neste exercício de conceitos básicos, você cria uma API sem servidor. As APIs sem servidor permitem que você se concentre em seus aplicativos, em vez de gastar tempo provisionando e gerenciando servidores. Este exercício leva menos de 20 minutos para ser concluído e é possível dentro do [nível gratuito da AWS](#).

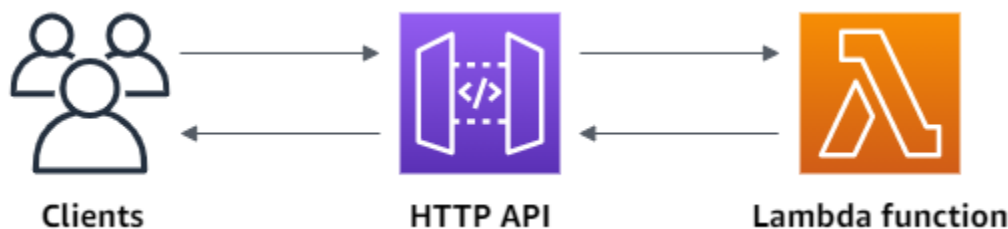
Primeiro, você cria uma função do Lambda usando o console do AWS Lambda. Em seguida, crie uma API HTTP usando o console do API Gateway. Em seguida, você invoca sua API.

Note

Este exercício usa uma API HTTP. O API Gateway também oferece suporte a APIs REST, que incluem mais recursos. Para ver um tutorial que usa uma API REST, consulte [the section called “Começar a usar o console da API REST”](#).

Para obter mais informações sobre a diferença entre as APIs HTTP e as APIs REST, consulte [the section called “Escolher entre APIs REST e APIs HTTP”](#).

Quando você invoca sua API HTTP, o API Gateway encaminha a solicitação para sua função do Lambda. O Lambda executa a função do Lambda e retorna uma resposta ao API Gateway. O API Gateway retorna uma resposta para você.



Para concluir esse exercício, você precisa de uma conta da AWS e de um usuário do AWS Identity and Access Management com acesso ao console. Para obter mais informações, consulte [Pré-requisitos](#).

Tópicos

- [Etapa 1: Criar uma função do Lambda](#)
- [Etapa 2: criar uma API HTTP](#)
- [Etapa 3: testar sua API](#)

- [\(Opcional\) Etapa 4: Limpar](#)
- [Próximas etapas](#)

Etapa 1: Criar uma função do Lambda

Você usa uma função do Lambda para o backend da sua API. O Lambda executa o código somente quando necessário e dimensiona automaticamente, desde algumas solicitações por dia a milhares por segundo.

Para este exemplo, você usa a função Node.js padrão do console do Lambda.

Como criar uma função do Lambda

1. Abra o console do Lambda em <https://console.aws.amazon.com/lambda>
2. Escolha Create function (Criar função).
3. Em Function name (Nome da função), insira **my-function**.
4. Em todas as outras opções, use a configuração padrão.
5. Escolha Create function (Criar função).

A função de exemplo retorna uma 200 resposta aos clientes e ao texto Hello from Lambda!.

Você pode modificar sua função do Lambda, desde que a resposta da função esteja alinhada com o [formato que o API Gateway requer](#).

O código de função padrão do Lambda deve ser semelhante ao seguinte:

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

Etapa 2: criar uma API HTTP

Em seguida, você cria uma API HTTP. O API Gateway também suporta APIs REST e APIs WebSocket, mas uma API HTTP é a melhor escolha para este exercício. As APIs REST oferecem

suporte a mais recursos do que as APIs HTTP, mas não precisamos desses recursos para este exercício. As APIs HTTP são projetadas com recursos mínimos para que possam ser oferecidas por um preço mais baixo. As APIs do WebSocket mantêm conexões persistentes com clientes para comunicação full-duplex, o que não é necessário para este exemplo.

A API HTTP fornece um endpoint de HTTP para sua função do Lambda. O API Gateway encaminha solicitações para sua função do Lambda e, em seguida, retorna a resposta da função aos clientes.

Para criar uma API HTTP

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Execute um destes procedimentos:
 - Para criar sua primeira API, para API HTTP, escolha Build (Criar).
 - Se você criou uma API antes, escolha Create API(Criar API) e, em seguida, escolha Build (Criar) para API HTTP.
3. Para Integrations (Integrações), escolha Add integration (Adicionar integração).
4. Escolha Lambda.
5. Em Lambda function (Função do Lambda), insira **my-function**.
6. Em API name (Nome da API), insira **my-http-api**.
7. Escolha Next (Próximo).
8. Revise a rota que o API Gateway cria para você e escolha Next (Avançar).
9. Revise o estágio criado pelo API Gateway para você e escolha Next (Avançar).
10. Escolha Create (Criar).

Agora você criou uma API HTTP com uma integração do Lambda que está pronta para receber solicitações de clientes.

Etapa 3: testar sua API

Em seguida, você testa sua API para se certificar de que ela está funcionando. Para simplificar, use um navegador da Web para invocar sua API.

Para testar sua API

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.

2. Selecione a API.
3. Observe o URL de invocação da sua API.

my-http-api

Stage: - ▼

Deploy

API details

Edit

API ID
abcd1234**Protocol**
HTTP**Created**
2023-10-30**Description**
My first HTTP API**Default endpoint**
✔ Enabled
<https://abcd1234.execute-api.us-west-1.amazonaws.com>**ARN**
[arn:aws:apigateway:us-west-1::apis/abcd1234](#)**Stages for my-http-api (1)**

Find resources

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

4. Copie o URL de invocação da sua API e insira-o em um navegador da Web. Anexe o nome da sua função do Lambda ao URL de chamada para chamar sua função do Lambda. Por padrão, o console do API Gateway cria uma rota com o mesmo nome da função do Lambda, my-function.

O URL deve ser semelhante a `https://abcdef123.execute-api.us-east-2.amazonaws.com/my-function`.

Seu navegador envia uma solicitação GET à API.

5. Verifique a resposta da sua API. Você deve ver o texto "Hello from Lambda!" no seu navegador.

(Opcional) Etapa 4: Limpar

Para evitar custos desnecessários, exclua os recursos que você criou como parte desse exercício de conceitos básicos. As etapas a seguir excluem sua API HTTP, sua função do Lambda e recursos associados.

Para excluir uma API HTTP

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.

2. Na página APIs , selecione uma API. Escolha Actions (Ações) e, depois, escolha Delete (Excluir).
3. Escolha Delete (Excluir).

Para excluir uma função do Lambda

1. Abra o console do Lambda em <https://console.aws.amazon.com/lambda>
2. Na página Functions(Funções), selecione uma função. Escolha Actions (Ações) e, depois, escolha Delete (Excluir).
3. Escolha Delete (Excluir).

Para excluir um grupo de logs de uma função do Lambda

1. No console do Amazon CloudWatch, abra a [página Log groups](#) (Grupos de log).
2. Na página Log groups (Grupos de log), selecione o grupo de log da função (/aws/lambda/my-function). Escolha Actions (Ações) e selecione Delete log group (Excluir grupo de log).
3. Escolha Delete (Excluir).

Para excluir a função de execução de uma função do Lambda

1. No console do AWS Identity and Access Management, abra a página [Roles](#) (Funções).
2. Selecione a atribuição da função, por exemplo, my-function-*31exxmpl*.
3. Selecione Delete role (Excluir função).
4. Escolha Sim, excluir.

Você pode automatizar a criação e a limpeza dos recursos da AWS usando o CloudFormation ou o AWS SAM. Para ver modelos de exemplo do CloudFormation, consulte [modelos de exemplo do CloudFormation](#).

Próximas etapas

Neste exemplo, você usou o Console de gerenciamento da AWS para criar uma API HTTP simples. A API HTTP invoca uma função do Lambda e retorna uma resposta aos clientes.

A seguir estão as próximas etapas à medida que você continua trabalhando com o API Gateway.

- [Configurar tipos adicionais de integrações de API](#), incluindo:
 - [Endpoints de HTTP](#)
 - [Recursos privados em uma VPC, como serviços do Amazon ECS](#)
 - [Serviços da AWS, como o Amazon Simple Queue Service e o Kinesis Data Streams AWS Step Functions](#)
- [Controlar acesso às suas APIs](#)
- [Habilitar registro para suas APIs](#)
- [Configurar o controle de utilização de suas APIs](#)
- [Configurar domínios personalizados de suas APIs](#)

Para obter ajuda com o Amazon API Gateway da comunidade, consulte o [Fórum de discussão do API Gateway](#). Ao entrar neste fórum, a AWS pode exigir que você faça login.

Para obter para o API Gateway diretamente da AWS, consulte as opções de suporte na página [Suporte da AWS](#).

Consulte também nossas [Perguntas frequentes](#) (FAQs) ou [entre em contato conosco diretamente](#).

Tutoriais e workshops do Amazon API Gateway

Os tutoriais e workshops a seguir fornecem exercícios práticos para ajudar você a aprender mais sobre o API Gateway. Para começar, recomendamos que você complete [the section called “Tutorial: Crie uma API REST com uma integração de proxy do Lambda”](#).

Tutoriais da API REST

- [Escolher um tutorial de integração do AWS Lambda](#)
- [Tutorial: Criar uma API REST importando um exemplo](#)
- [Escolher um tutorial de integração HTTP](#)
- [Tutorial: Crie uma API REST com uma integração privada](#)
- [Tutorial: Crie uma API REST com uma integração da AWS](#)
- [Tutorial: Criar uma API REST de calculadora com duas integrações de serviços da AWS e uma integração sem proxy do Lambda](#)
- [Tutorial: Crie uma API REST como um proxy do Amazon S3](#)
- [Tutorial: Criar uma API REST como um proxy do Amazon Kinesis](#)
- [Tutorial: como criar uma API REST usando SDKs da AWS ou a AWS CLI](#)
- [Tutorial: Crie uma API REST privada](#)

Tutoriais da API HTTP

- [Tutorial: Crie uma API HTTP CRUD com o Lambda e o DynamoDB](#)
- [Tutorial: Crie uma API HTTP com uma integração privada a um serviço do Amazon ECS](#)

Tutoriais da API WebSocket

- [Tutorial: Crie uma aplicação de bate-papo de WebSocket com uma API de WebSocket, o Lambda e o DynamoDB](#)

Workshops

- [Crie uma aplicação Web sem servidor](#)
- [CI/CD para aplicações sem servidor](#)

- [Workshop de segurança sem servidor](#)
- [Gerenciamento, autenticação e autorização de identidade sem servidor](#)
- [The Amazon API Gateway Workshop](#)

Tutoriais da API REST do Amazon API Gateway

Os tutoriais a seguir fornecem exercícios práticos para ajudar você a saber mais sobre as APIs REST do API Gateway.

Tópicos

- [Escolher um tutorial de integração do AWS Lambda](#)
- [Tutorial: Criar uma API REST importando um exemplo](#)
- [Escolher um tutorial de integração HTTP](#)
- [Tutorial: Crie uma API REST com uma integração privada](#)
- [Tutorial: Crie uma API REST com uma integração da AWS](#)
- [Tutorial: Criar uma API REST de calculadora com duas integrações de serviços da AWS e uma integração sem proxy do Lambda](#)
- [Tutorial: Crie uma API REST como um proxy do Amazon S3](#)
- [Tutorial: Criar uma API REST como um proxy do Amazon Kinesis](#)
- [Tutorial: como criar uma API REST usando SDKs da AWS ou a AWS CLI](#)
- [Tutorial: Crie uma API REST privada](#)

Escolher um tutorial de integração do AWS Lambda

Para criar uma API com integrações do Lambda, é possível usar a integração de proxy ou sem proxy do Lambda.

Com a integração de proxy do Lambda, a entrada da função do Lambda pode ser expressa como qualquer combinação de cabeçalhos de solicitação, variáveis de caminho, parâmetros de strings de consulta, corpo e dados de configuração de API. Só é necessário escolher uma função do Lambda. O API Gateway configura a solicitação de integração e a resposta de integração para você. Após a configuração, o método de API poderá evoluir sem modificar as configurações existentes. Isso é possível porque a função do Lambda de backend analisa os dados da solicitação recebida e responde ao cliente.

Na integração sem proxy do Lambda, é necessário garantir que a entrada da função do Lambda seja fornecida como a carga da solicitação de integração. Você deve mapear todos os dados de entrada fornecidos pelo cliente como parâmetros da solicitação no corpo da solicitação de integração apropriada. Também pode ser necessário converter o corpo da solicitação fornecida pelo cliente em um formato reconhecido pela função do Lambda.

Nas integrações com ou sem proxy do Lambda, é possível usar uma função do Lambda em uma conta diferente da conta em que criou a API.

Tópicos

- [Tutorial: Crie uma API REST com uma integração de proxy do Lambda](#)
- [Tutorial: Crie uma API REST com uma integração de não proxy do Lambda](#)
- [Tutorial: Crie uma API REST com uma integração de proxy do Lambda entre contas](#)

Tutorial: Crie uma API REST com uma integração de proxy do Lambda

A [integração de proxy do Lambda](#) é um tipo de integração leve e flexível da API do API Gateway que permite integrar um método de API ou uma API inteira a uma função do Lambda. A função do Lambda pode ser escrita em [qualquer linguagem compatível com o Lambda](#). Por se tratar de uma integração de proxy, você pode alterar a implementação da função do Lambda a qualquer momento, sem necessidade de implantar sua API novamente.

Neste tutorial, você faz o seguinte:

- Criar um "Hello, World!" Função do Lambda para ser o backend da API.
- Criar e testar um "Hello, World!" API com integração de proxy do Lambda.

Tópicos

- [Criar um "Hello, World!" Função do Lambda](#)
- [Criar um "Hello, World!" API](#)
- [Implantar e testar a API](#)

Criar um "Hello, World!" Função do Lambda

Como criar um "Hello, World!" Função do Lambda no console do Lambda

1. Abra o console do Lambda em <https://console.aws.amazon.com/lambda>

2. Na barra de navegação da AWS, escolha uma [Região da AWS](#).

 Note

Anote a região na qual você criará a função do Lambda. Você precisará dela ao criar a API.

3. Selecione Functions (Funções) no painel de navegação.
4. Escolha Create function (Criar função).
5. Escolha Author from scratch (Criar do zero).
6. Em Basic information (Informações básicas), faça o seguinte:
 - a. Em Function name (Nome da função), insira **GetStartedLambdaProxyIntegration**.
 - b. Em Tempo de execução, escolha o último runtime Node.js ou Python compatível.
 - c. Em Arquitetura, mantenha a configuração padrão.
 - d. Em Permissions (Permissões), expanda Change default execution role (Alterar função de execução padrão). Na lista suspensa Perfil de execução, escolha Criar perfil com base em modelos de política da AWS.
 - e. Em Role name (Nome da função), insira **GetStartedLambdaBasicExecutionRole**.
 - f. Deixe em branco o campo Policy templates (Modelos de política).
 - g. Escolha Create function (Criar função).
7. Em Function code (Código de função), no editor de código em linha, copie/cole o seguinte código:

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. Escolha Deploy (Implantar).

Criar um "Hello, World!" API

Agora crie uma API para “Hello, World!” Função do Lambda usando o console do API Gateway.

Como criar um “Hello, World!” solicitações de

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Se esta for a primeira vez que você usa o API Gateway, você verá uma página com os recursos do serviço. Em REST API, escolha Build (Criar). Quando o pop-up Create Example API (Criar API de exemplo) for exibido, escolha OK.

Se essa não for a primeira vez que você usa o API Gateway, escolha Create API (Criar API). Em REST API, escolha Build (Criar).

3. Em API name (Nome da API), insira **LambdaProxyAPI**.
4. (Opcional) Em Description (Descrição), insira uma descrição.
5. Mantenha Tipo de endpoint de API definido como Regional.
6. Em Tipo de endereço IP, selecione IPv4.
7. Selecione Criar API.

Depois de criar uma API, você criará um recurso. Normalmente, os recursos da API são organizados em uma árvore de recursos, de acordo com a lógica do aplicativo. Neste exemplo, você criará um recurso `/helloworld`.

Para criar um recurso

1. Selecione Criar recurso.
2. Mantenha Recurso proxy desativado.
3. Mantenha Caminho do recurso como `/`.
4. Em Resource Name (Nome do recurso), insira **helloworld**.
5. Mantenha CORS (Compartilhamento de recursos de origem cruzada) desativado.
6. Selecione Criar recurso.

Em uma integração de proxy, a solicitação inteira é enviada à função do Lambda de backend como está, por meio de um método ANY genérico que representa qualquer método HTTP. O método HTTP real é especificado pelo cliente no tempo de execução. O método ANY permite usar uma única configuração do método da API para todos os métodos HTTP compatíveis: DELETE, GET, HEAD, OPTIONS, PATCH, POST e PUT.

Como criar um método **ANY**

1. Selecione o recurso `/helloworld` e, depois, Criar método.
2. Em Tipo de método, selecione ANY.
3. Em Tipo de integração, selecione Função do Lambda.
4. Ative Integração de proxy do Lambda.
5. Em Função do Lambda, selecione a Região da AWS onde você criou a função do Lambda e, depois, insira o nome da função.

6. Para usar o valor de tempo limite padrão de 29 segundos, deixe a opção Tempo limite padrão ativada. Para definir um tempo limite personalizado, selecione Tempo limite padrão e insira um valor de tempo limite entre 50 e 29000 milissegundos.
7. Escolha Criar método.

Implantar e testar a API

Para implantar sua API

1. Escolha Implantar API.
2. Em Estágio, selecione Novo estágio.
3. Em Stage name (Nome do estágio), insira **test**.
4. (Opcional) Em Description (Descrição), insira uma descrição.
5. Escolha Implantar.
6. Em Detalhes do estágio, escolha o ícone de cópia para copiar o URL de invocação da API.

Usar o navegador e cURL para testar uma API com integração de proxy do Lambda

Você pode usar um navegador ou [cURL](#) para testar sua API.

Para testar solicitações GET usando apenas parâmetros de string de consulta, você pode inserir o URL do recurso helloworld da API em uma barra de endereço do navegador.

Para criar o URL do recurso helloworld da API, acrescente o recurso helloworld e o parâmetro da string de consulta ?greeter=John ao URL de invocação. Seu URL deve ter uma aparência semelhante ao seguinte.

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

Para outros métodos, é necessário usar utilitários de teste de API REST mais avançados, como o [POSTMAN](#) ou o [cURL](#). Este tutorial usa cURL. Os exemplos de comando cURL a seguir pressupõem que cURL foi instalado em seu computador.

Para testar a API implantada usando cURL:

1. Abra uma janela do terminal.

2. Copie o comando cURL a seguir e cole-o na janela do terminal, substituindo o URL de invocação pelo que foi copiado na etapa anterior e adicione **/helloworld** ao final do URL.

 Note

Se você estiver executando o comando no Windows, use a seguinte sintaxe:

```
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 chamar a API com o parâmetro de string de consulta de ?greeter=John:

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

- b. Para chamar a API com um parâmetro de cabeçalho 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. Como chamar a API com um corpo 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" }'
```

Em todos os casos, a saída é uma resposta 200 com o corpo de resposta a seguir:

```
Hello, John!
```

Tutorial: Crie uma API REST com uma integração de não proxy do Lambda

Nesta demonstração, usamos o console do API Gateway para criar uma API que permite a um cliente chamar funções do Lambda por meio da integração não proxy do Lambda (também conhecida como integração personalizada). Para obter mais informações sobre AWS Lambda e as funções do Lambda, consulte o [Guia do desenvolvedor do AWS Lambda](#).

Para facilitar o aprendizado, escolhemos uma função do Lambda simples com configuração mínima da API para orientar você pelas etapas de criação de uma API do API Gateway com a integração personalizada do Lambda. Quando necessário, descrevemos parte da lógica. Para ver um exemplo mais detalhado da integração personalizada do Lambda, consulte [Tutorial: Criar uma API REST de calculadora com duas integrações de serviços da AWS e uma integração sem proxy do Lambda](#).

Antes de criar a API, configure o backend do Lambda criando uma função do Lambda no AWS Lambda, conforme descrito a seguir.

Tópicos

- [Criar uma função do Lambda para integração não proxy do Lambda](#)
- [Criar uma API com integração não proxy do Lambda](#)
- [Testar a chamada do método de API](#)
- [Implantar a API](#)
- [Testar a API em uma etapa de implantação](#)
- [Limpar](#)

Criar uma função do Lambda para integração não proxy do Lambda

Note

A criação de funções do Lambda pode resultar em cobranças na conta da AWS.

Nesta etapa, crie uma função do Lambda no estilo “Hello, World!” para a integração personalizada do Lambda. Durante esta demonstração, a função é chamada `GetStartedLambdaIntegration`.

A implementação dessa função do Lambda `GetStartedLambdaIntegration` é a seguinte:

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 a integração personalizada do Lambda o API Gateway transmite a entrada para a função do Lambda do cliente como o corpo da solicitação de integração. O objeto event do manipulador da função do Lambda é a entrada.

Nossa função do Lambda é simples. Ela analisa o objeto de entrada event para as propriedades name, city, time e day. Em seguida, ela retorna uma saudação, como um objeto JSON de {"message":greeting}, para o autor da chamada. A mensagem é no padrão "Good [morning|afternoon|day], [*name*|you] in [*city*|World]. Happy *day*!". Presume-se que a entrada para a função do Lambda seja um dos seguintes objetos JSON:

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

```
"name" : "..."  
}
```

Para obter mais informações, consulte o [Guia do desenvolvedor do AWS Lambda](#).

Além disso, a função registra sua execução no Amazon CloudWatch chamando `console.log(...)`. Isso é útil para rastrear as chamadas ao depurar a função. Para permitir que a função `GetStartedLambdaIntegration` registre a chamada, defina uma função do IAM com as políticas apropriadas para que a função do Lambda crie os streams do CloudWatch e adicione entradas de log aos streams. O console do Lambda orienta você na criação das funções e políticas do IAM necessárias.

Se você configurar a API sem usar o console do API Gateway, tal como ao [importar uma API de um arquivo do OpenAPI](#), é necessário criar explicitamente, se necessário, e configurar uma função e política de invocação para que o API Gateway invoque as funções do Lambda. Para obter mais informações sobre como configurar funções de invocação e execução do Lambda para uma API do API Gateway, consulte [Controlar o acesso a uma API REST com permissões do IAM](#).

Em comparação com `GetStartedLambdaProxyIntegration`, a função do Lambda para a integração de proxy do Lambda, a função `GetStartedLambdaIntegration` do Lambda para a integração personalizada do Lambda recebe apenas a entrada do corpo da solicitação de integração da API do API Gateway. A função pode retornar uma saída de qualquer objeto JSON, uma string, um número, um Booleano ou até mesmo um blob binário. Em contrapartida, a função do Lambda para a integração de proxy do Lambda pode aceitar a entrada de quaisquer dados da solicitação, mas deve retornar uma saída de determinado objeto JSON. A função `GetStartedLambdaIntegration` para a integração personalizada do Lambda pode ter os parâmetros de solicitação de API como entrada, desde que o API Gateway mapeie os parâmetros de solicitação de API necessários para o corpo da solicitação de integração antes de encaminhar a solicitação do cliente ao backend. Para que isso aconteça, o desenvolvedor da API deve criar um modelo de mapeamento e configurá-lo no método de API ao criar a API.

Agora, crie a função do Lambda `GetStartedLambdaIntegration`.

Como criar a função do Lambda **`GetStartedLambdaIntegration`** para integração personalizada do Lambda

1. Abra o console do AWS Lambda em <https://console.aws.amazon.com/lambda/>.
2. Execute um destes procedimentos:

- Se a página de boas-vindas for exibida, escolha Get Started Now (Começar a usar agora) e Create function (Criar função).
 - Se a página da lista Lambda > Functions (Lambda > Funções) for exibida, escolha Create function (Criar função).
3. Escolha Author from scratch (Criar do zero).
 4. Na tela Author from scratch (Criar do zero), faça o seguinte:
 - a. Em Name (Nome), insira **GetStartedLambdaIntegration** como o nome da função do Lambda.
 - b. Em Tempo de execução, escolha o último runtime Node.js ou Python compatível.
 - c. Em Arquitetura, mantenha a configuração padrão.
 - d. Em Permissions (Permissões), expanda Change default execution role (Alterar função de execução padrão). Na lista suspensa Perfil de execução, escolha Criar perfil com base em modelos de política da AWS.
 - e. Para Role name (Nome da função), insira um nome para sua função (por exemplo, **GetStartedLambdaIntegrationRole**).
 - f. Para Policy templates (Modelos de política), escolha Simple microservice permissions (Permissões de microserviço simples).
 - g. Escolha Create function (Criar função).
 5. No painel Configure function (Configurar função), em Function code (Código de função) faça o seguinte:
 - a. Copie o código da função do Lambda listado no início desta seção e cole-o no editor de código em linha.
 - b. Deixe as opções padrão para todos os outros campos nesta seção.
 - c. Escolha Deploy (Implantar).
 6. Para testar a função recém-criada, escolha a guia Teste.
 - a. Em Nome do evento, insira **HelloWorldTest**.
 - b. Para JSON do evento, substitua o código padrão pelo seguinte.

```
{
  "name": "Jonny",
  "city": "Seattle",
  "time": "morning",
```

```
"day": "Wednesday"
}
```

- c. Escolha Test (Testar) para invocar a função. A seção Execution result: succeeded (Resultada da execução: bem-sucedida) é exibida. Expanda Detalhes veja a saída a seguir.

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

A saída também é gravada no CloudWatch Logs.

Como exercício extra, você pode usar o console do IAM para visualizar a função do IAM (GetStartedLambdaIntegrationRole) criada como parte da criação da função do Lambda. Há duas políticas em linha anexadas à esta função do IAM. Uma estipula as permissões mais básicas para execução do Lambda. Ela permite chamar o CreateLogGroup do CloudWatch para quaisquer recursos do CloudWatch de sua conta na região onde a função do Lambda for criada. Essa política também permite a criação de streams e eventos de registro em log do CloudWatch para a função do 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:*"
      ]
    }
  ]
}
```

```
    }  
  ]  
}
```

O outro documento de política se aplica à invocação de outro serviço da AWS que não é usado neste exemplo. Você pode ignorá-lo por enquanto.

Associada à função do IAM, há uma entidade confiável, que é `lambda.amazonaws.com`. Esta é a relação de confiança:

JSON

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

A combinação dessa relação de confiança e da política em linha possibilita que a função do Lambda invoque uma função `console.log()` para registro de eventos no CloudWatch Logs.

Criar uma API com integração não proxy do Lambda

Com a função do Lambda (`GetStartedLambdaIntegration`) criada e testada, você está pronto para expor a função por meio de uma API do API Gateway. Para fins de ilustração, a função do Lambda está exposta com um método HTTP genérico. Usamos o corpo de solicitação, uma variável do caminho URL, uma string de consulta e um cabeçalho para receber os dados de entrada necessários do cliente. Ativamos o validador de solicitações do API Gateway para a API a fim de garantir que todos os dados necessários sejam definidos e especificados apropriadamente. Configuramos um modelo para mapeamento para o API Gateway para transformar os dados de solicitação fornecidos pelo cliente no formato válido, conforme exigido pela função de backend do Lambda.

Como criar uma API com uma integração não proxy do Lambda

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Se esta for a primeira vez que você usa o API Gateway, você verá uma página com os recursos do serviço. Em REST API, escolha Build (Criar). Quando o pop-up Create Example API (Criar API de exemplo) for exibido, escolha OK.

Se essa não for a primeira vez que você usa o API Gateway, escolha Create API (Criar API). Em REST API, escolha Build (Criar).

3. Em API name (Nome da API), insira **LambdaNonProxyAPI**.
4. (Opcional) Em Description (Descrição), insira uma descrição.
5. Mantenha Tipo de endpoint de API definido como Regional.
6. Em Tipo de endereço IP, selecione IPv4.
7. Selecione Criar API.

Depois de criar uma API, você criará um recurso `/city`. Este é um exemplo de recurso com uma variável de caminho que recebe uma entrada do cliente. Posteriormente, você vai associar essa variável de caminho à entrada da função do Lambda usando um modelo de mapeamento.

Para criar um recurso

1. Selecione Criar recurso.
2. Mantenha Recurso proxy desativado.
3. Mantenha Caminho do recurso como `/`.
4. Em Resource Name (Nome do recurso), insira **`{city}`**.
5. Mantenha CORS (Compartilhamento de recursos de origem cruzada) desativado.
6. Selecione Criar recurso.

Depois de criar um recurso `/city`, você criará um método ANY. O verbo HTTP ANY é um espaço reservado para um método HTTP válido que um cliente envia na ocasião da execução. Este exemplo mostra que o método ANY pode ser usado para integração personalizada do Lambda assim como para integração de proxy do Lambda.

Como criar um método **ANY**

1. Selecione o recurso `/city` e, depois, Criar método.

2. Em Tipo de método, selecione ANY.
3. Em Tipo de integração, selecione Função do Lambda.
4. Mantenha a opção Integração do proxy do Lambda desativada.
5. Em Função do Lambda, selecione a Região da AWS onde você criou a função do Lambda e, depois, insira o nome da função.
6. Escolha Configurações de solicitação de método.

Agora, ative um validador de solicitações para uma variável de caminho de URL, um parâmetro de string de consulta e um cabeçalho a fim de garantir que todos os dados necessários sejam definidos. Neste exemplo, você vai criar um parâmetro de string de consulta `time` e um cabeçalho `day`.

7. Em Validador de solicitação, selecione Validar parâmetros de string de consulta e cabeçalhos.
8. Selecione Parâmetros de string de consulta de URL e faça o seguinte:
 - a. Escolha Add query string (Adicionar string de consulta).
 - b. Em Nome, digite **time**.
 - c. Ative a opção Obrigatório.
 - d. Mantenha Armazenamento em cache desativado.
9. Selecione Cabeçalhos de solicitação HTTP e faça o seguinte:
 - a. Escolha Add header (Adicionar cabeçalho).
 - b. Em Nome, digite **day**.
 - c. Ative a opção Obrigatório.
 - d. Mantenha Armazenamento em cache desativado.
10. Escolha Criar método.

Depois de ativar um validador de solicitações, você vai configurar a solicitação de integração do método ANY adicionando um modelo de mapeamento de corpo para transformar a solicitação recebida em uma carga útil do JSON, conforme exigido pela função do Lambda de back-end.

Como configurar a solicitação de integração

1. Na guia Solicitação de integração em Configurações de solicitação de integração, selecione Editar.

2. Em Passagem do corpo da solicitação, selecione Quando não há modelos definidos (recomendado).
3. Selecione Modelos de mapeamento.
4. Escolha Add mapping template (Adicionar modelo de mapeamento).
5. Em Tipo de conteúdo, insira **application/json**.
6. Em Corpo do modelo, insira o seguinte código:

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

7. Escolha Salvar.

Testar a chamada do método de API

O console do API Gateway fornece uma instalação de testes para que você teste a invocação à API antes que ela seja implantada. Você pode usar o recurso de teste do console para testar a API enviando a seguinte solicitação:

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

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

Nesta solicitação de teste, você definirá ANY como POST, definirá {city} como Seattle, atribuirá Wednesday como o valor de cabeçalho day e atribuirá "John" como o valor callerName.

Como testar o método **ANY**

1. Selecione a guia Testar. Talvez seja necessário selecionar o botão de seta para a direita para mostrar a guia.
2. Em Tipo de método, selecione POST.
3. Em Caminho, em cidade, insira **Seattle**.

4. Em Strings de consulta, digite **time=morning**.
5. Em Cabeçalhos, insira **day:Wednesday**.
6. Em Corpo da solicitação, insira **{ "callerName": "John" }**.
7. Escolha Test (Testar).

Verifique se a carga de resposta retornada é como se segue:

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

Também é possível visualizar os logs para examinar como o API Gateway processa a solicitação e a resposta.

Execution log for request test-request

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

Thu Aug 31 01:07:25 UTC 2017 : HTTP Method: POST, Resource Path: /Seattle

Thu Aug 31 01:07:25 UTC 2017 : Method request path: {city=Seattle}

Thu Aug 31 01:07:25 UTC 2017 : Method request query string: {time=morning}

Thu Aug 31 01:07:25 UTC 2017 : Method request headers: {day=Wednesday}

Thu Aug 31 01:07:25 UTC 2017 : Method request body before transformations:

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

Thu Aug 31 01:07:25 UTC 2017 : Request validation succeeded for content type application/json

Thu Aug 31 01:07:25 UTC 2017 : Endpoint request URI: https://

lambda.us-west-2.amazonaws.com/2015-03-31/functions/arn:aws:lambda:us-west-2:123456789012:function:GetStartedLambdaIntegration/invocations

Thu Aug 31 01:07:25 UTC 2017 : Endpoint request headers: {x-amzn-lambda-integration-tag=test-request,

Authorization=*****

X-Amz-Date=20170831T010725Z, x-amzn-apigateway-api-id=beags1mnid, X-Amz-Source-Arn=arn:aws:execute-api:us-west-2:123456789012:beags1mnid/null/POST/{city}, Accept=application/json, User-Agent=AmazonAPIGateway_beags1mnid,

X-Amz-Security-Token=FQoDYXdzELL////////wEaDMHGzEdEOT/VvGhabiK3AzgKrJw+3zLqJZG4Ph0q12K6W21+QotY2rrZy0zqhLoiuRg3CAYNQ2eqgL5D54+63ey9bIdtwHGoyBdq8ecWxJK/YUnT2Rau0L9HCG5p7FC05h3IvwlFfvcidQNXeYvsKJTLXI05/

yEnY3ttIANpNYL0ezD9Es8rBfyruHfJf0qextKlsC8DymCcqlGkig8qLKcZ0hWJWwiPJiFgL7laabXs++ZhCa4hdZo4iq1G729DE4gaV1mJVdoAagIUwLMo+y4NxFDu0r7I0/

E05nYcCrrpGVVBYiGk7H4T6sXuhTkbnNqVmXtV3ch5b0lh7 [TRUNCATED]

Thu Aug 31 01:07:25 UTC 2017 : Endpoint request body after transformations: {

```

    "city": "Seattle",
    "time": "morning",
    "day": "Wednesday",
    "name" : "John"
  }
Thu Aug 31 01:07:25 UTC 2017 : Sending request to https://lambda.us-
west-2.amazonaws.com/2015-03-31/functions/arn:aws:lambda:us-
west-2:123456789012:function:GetStartedLambdaIntegration/invocations
Thu Aug 31 01:07:25 UTC 2017 : Received response. Integration latency: 328 ms
Thu Aug 31 01:07:25 UTC 2017 : Endpoint response body before transformations:
{"greeting":"Good morning, John of Seattle. Happy Wednesday!"}
Thu Aug 31 01:07:25 UTC 2017 : Endpoint response headers: {x-amzn-Remapped-Content-
Length=0, x-amzn-RequestId=c0475a28-8de8-11e7-8d3f-4183da788f0f, Connection=keep-
alive, Content-Length=62, Date=Thu, 31 Aug 2017 01:07:25 GMT, X-Amzn-Trace-
Id=root=1-59a7614d-373151b01b0713127e646635;sampld=0, Content-Type=application/json}
Thu Aug 31 01:07:25 UTC 2017 : Method response body after transformations:
{"greeting":"Good morning, John of Seattle. Happy Wednesday!"}
Thu Aug 31 01:07:25 UTC 2017 : Method response headers: {X-Amzn-Trace-
Id=sampld=0;root=1-59a7614d-373151b01b0713127e646635, Content-Type=application/json}
Thu Aug 31 01:07:25 UTC 2017 : Successfully completed execution
Thu Aug 31 01:07:25 UTC 2017 : Method completed with status: 200

```

Os logs mostram a solicitação recebida antes do mapeamento e a solicitação de integração após o mapeamento. Quando um teste falha, os logs são úteis para avaliar se a entrada original está correta ou se o modelo de mapeamento funciona corretamente.

Implantar a API

A invocação de teste é uma simulação e tem limitações. Por exemplo, ela ignora qualquer mecanismo de autorização promulgado na API. Para testar a execução da API em tempo real, você deve implantar a API primeiro. Para implantar uma API, você cria um estágio para criar um snapshot da API naquele momento. O nome do estágio também define o caminho base após o nome de host padrão da API. O recurso raiz da API é anexado após o nome do estágio. Quando você modifica a API, deve reimplantá-la em um estágio novo ou existente antes que as alterações entrem em vigor.

Para implantar a API em um estágio

1. Escolha Implantar API.
2. Em Estágio, selecione Novo estágio.
3. Em Stage name (Nome do estágio), insira **test**.

 Note

A entrada deve ser texto codificado UTF-8 (ou seja, não localizado).

4. (Opcional) Em Description (Descrição), insira uma descrição.
5. Escolha Implantar.

Em Detalhes do estágio, escolha o ícone de cópia para copiar o URL de invocação da API. O padrão geral do URL básico da API é `https://api-id.region.amazonaws.com/stageName`. Por exemplo, o URL básico da API (beags1mnid) criada na região us-west-2 e implantada no estágio test é `https://beags1mnid.execute-api.us-west-2.amazonaws.com/test`.

Testar a API em uma etapa de implantação

Há várias maneiras para testar uma API implantada. Para solicitações GET usando apenas variáveis de caminho do URL ou parâmetros de strings de consulta, é possível digitar o URL de recurso da API em um navegador. Para outros métodos, é necessário usar utilitários de teste de API REST mais avançados, como o [POSTMAN](#) ou o [cURL](#).

Para testar a API usando cURL

1. Abra uma janela de terminal em seu computador local conectado à Internet.
2. Para testar POST /Seattle?time=evening:

Copie o seguinte comando cURL e cole-o na janela do 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"
}'
```

Você receberá uma resposta bem-sucedida com a seguinte carga:

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

Se você alterar POST para PUT nesta solicitação de método, obterá a mesma resposta.

Limpar

Se você não precisar mais das funções do Lambda criadas para esta demonstração, poderá excluí-las agora. Você também pode excluir os recursos do IAM que o acompanham.

Warning

Se você pretende completar as outras demonstrações desta série, não exclua a função de execução Lambda ou a função de invocação do Lambda. Se você excluir uma função do Lambda da qual as suas APIs dependem, essas APIs deixarão de funcionar. A exclusão de uma função do Lambda não pode ser desfeita. Se quiser usar a função do Lambda novamente, você deverá recriar essa função.

Se você excluir um recurso do IAM do qual depende uma função do Lambda, esta última deixará de funcionar, juntamente com as APIs que dependem dessa função. A exclusão de um recurso do IAM não pode ser desfeita. Se quiser usar o recurso do IAM novamente, você deverá recriá-lo.

Como excluir a função do Lambda

1. Faça login no Console de gerenciamento da AWS e abra o console AWS Lambda em <https://console.aws.amazon.com/lambda/>.
2. Na lista de funções, escolha GetStartedLambdaIntegration, Ações e Excluir função. Quando solicitado, escolha Delete (Excluir) novamente.

Como excluir os recursos do IAM associados

1. Abra o console do IAM em <https://console.aws.amazon.com/iam/>.
2. Em Details (Detalhes), escolha Roles (Funções).
3. Na lista de funções, escolha GetStartedLambdaIntegrationRole, Ações da função e Excluir função. Siga as etapas no console para excluir o perfil.

Tutorial: Crie uma API REST com uma integração de proxy do Lambda entre contas

Agora você pode usar a função do AWS Lambda de uma conta da AWS diferente como seu backend de integração da API. Cada conta pode estar em qualquer região onde o Amazon API Gateway está disponível. Isso facilita o gerenciamento centralizado e o compartilhamento das funções do Lambda de backend em várias APIs.

Nesta seção, mostramos como configurar a integração de proxy do Lambda entre contas usando o console do Amazon API Gateway.

Criar uma API para integração do Lambda entre contas do API Gateway

Como criar uma API

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Se esta for a primeira vez que você usa o API Gateway, você verá uma página com os recursos do serviço. Em REST API, escolha Build (Criar). Quando o pop-up Create Example API (Criar API de exemplo) for exibido, escolha OK.

Se essa não for a primeira vez que você usa o API Gateway, escolha Create API (Criar API). Em REST API, escolha Build (Criar).

3. Em API name (Nome da API), insira **CrossAccountLambdaAPI**.
4. (Opcional) Em Description (Descrição), insira uma descrição.
5. Mantenha Tipo de endpoint de API definido como Regional.
6. Em Tipo de endereço IP, selecione IPv4.
7. Selecione Criar API.

Criar a função de integração do Lambda em outra conta

Agora você criará uma função do Lambda em uma conta diferente daquela em que criou a API demonstrativa.

Criar uma função do Lambda em outra conta

1. Faça login no console do Lambda usando uma conta diferente daquela em que você criou sua API do API Gateway.
2. Escolha Create function (Criar função).
3. Escolha Author from scratch (Criar do zero).

4. Em Author from scratch (Criar do zero), faça o seguinte:
 - a. Em Function name (Nome da função), insira um nome.
 - b. Na lista suspensa Runtime (Tempo de execução), escolha um tempo de execução Node.js compatível.
 - c. Em Arquitetura, mantenha a configuração padrão.
 - d. Em Permissions (Permissões), expanda Choose or create an execution role (Escolher ou criar uma função de execução). É possível criar uma função ou escolher uma função existente.
 - e. Escolha Create function (Criar função) para continuar.
5. Role para baixo, até o painel Function code (Código da função).
6. Insira a implementação da função Node.js de [the section called “Tutorial: Crie uma API REST com uma integração de proxy do Lambda”](#).
7. Escolha Deploy (Implantar).
8. Anote o ARN completo da sua função (no canto superior direito do painel de funções do Lambda). Você precisará dele ao criar a integração do Lambda entre contas.

Configurar a integração do Lambda entre contas

Depois de ter uma função de integração do Lambda em uma conta diferente, você poderá usar o console do API Gateway para adicioná-la à sua API na sua primeira conta.

Note

Se você estiver configurando um autorizador entre regiões e entre contas, o `sourceArn` que é adicionado à função de destino deve usar a região da função, e não a região da API.

Depois de criar uma API, você criará um recurso. Normalmente, os recursos da API são organizados em uma árvore de recursos, de acordo com a lógica do aplicativo. Neste exemplo, você criará um recurso `/helloworld`.

Para criar um recurso

1. Selecione Criar recurso.
2. Mantenha Recurso proxy desativado.

3. Mantenha Caminho do recurso como `/`.
4. Em Resource Name (Nome do recurso), insira **helloworld**.
5. Mantenha CORS (Compartilhamento de recursos de origem cruzada) desativado.
6. Selecione Criar recurso.

Depois de criar um recurso, você criará um método GET. Você vai integrar o método GET a uma função do Lambda em outra conta.

Como criar um método **GET**

1. Selecione o recurso `/helloworld` e, depois, Criar método.
2. Em Tipo de método, selecione GET.
3. Em Tipo de integração, selecione Função do Lambda.
4. Ative Integração de proxy do Lambda.
5. Em Função do Lambda, insira o ARN completo da função do Lambda na Etapa 1.

No console do Lambda, você pode encontrar o ARN da sua função no canto superior direito da janela.

6. Ao inserir o ARN, uma string de comando `aws lambda add-permission` é exibida. Esta política concederá acesso da sua primeira conta à função do Lambda da segunda conta. Copie e cole a string de comando `aws lambda add-permission` em uma janela da AWS CLI configurada para a segunda conta.
7. Escolha Criar método.

É possível ver a política atualizada da função no console do Lambda.

(Opcional) Como ver a política atualizada

1. Faça login no Console de gerenciamento da AWS e abra o console do AWS Lambda em <https://console.aws.amazon.com/lambda/>.
2. Selecione a função do Lambda.
3. Escolha Permissions (Permissões).

É necessário ver uma política Allow com uma cláusula Condition na qual o `AWS:SourceArn` é o ARN para o método GET da sua API.

Tutorial: Criar uma API REST importando um exemplo

Você pode usar o console do Amazon API Gateway para criar e testar uma API REST simples com a integração HTTP de um site PetStore. A definição de API é pré-configurada como um arquivo OpenAPI 2.0. Depois de carregar a definição da API no API Gateway, é possível usar o console do API Gateway para examinar a estrutura básica da API ou simplesmente implantar e testar a API.

A API PetStore de exemplo oferece suporte aos métodos a seguir para que um cliente acesse o site de backend HTTP de `http://petstore-demo-endpoint.execute-api.com/petstore/pets`.

Note

Esse tutorial usa um endpoint HTTP como exemplo. Ao criar suas próprias APIs, recomendamos que você use endpoints HTTPS para as integrações HTTP.

- GET `/`: para acesso de leitura do recurso raiz da API que não está integrado a qualquer endpoint de backend. O API Gateway responde com uma visão geral do site PetStore. Este é um exemplo do tipo de integração MOCK.
- GET `/pets`: para acesso de leitura ao recurso `/pets` da API que está integrado ao recurso de backend `/pets` de nome idêntico. O backend retorna uma página de animais de estimação disponíveis na PetStore. Este é um exemplo do tipo de integração HTTP. O URL do endpoint de integração é `http://petstore-demo-endpoint.execute-api.com/petstore/pets`.
- POST `/pets`: para acesso de gravação ao recurso `/pets` da API que está integrado ao recurso de backend `/petstore/pets`. Depois de receber uma solicitação correta, o backend adiciona o animal de estimação especificado à PetStore e retorna o resultado para o autor da chamada. A integração também é HTTP.
- GET `/pets/{petId}`: para acesso de leitura a um animal de estimação conforme identificado por um valor `petId` especificado como uma variável de caminho do URL da solicitação recebida. Este método também tem o tipo de integração HTTP. O backend retorna o animal de estimação especificado encontrado na PetStore. O URL do endpoint HTTP de backend é `http://petstore-demo-endpoint.execute-api.com/petstore/pets/n`, onde *n* é um número inteiro como o identificador do animal de estimação consultado.

A API oferece suporte ao acesso de CORS através dos métodos OPTIONS do tipo de integração MOCK. O API Gateway retorna os cabeçalhos necessários para suporte ao acesso de CORS.

O procedimento a seguir descreve as etapas para criar e testar uma API a partir de um exemplo usando o console do API Gateway.

Para importar, criar e testar a API de exemplo

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Execute um destes procedimentos:
 - Para criar sua primeira API, em API REST, escolha Criar.
 - Se você criou uma API antes, escolha Criar API e, depois, escolha Criar para API REST.
3. Em Criar API REST, selecione API de exemplo e escolha Importar para criar a API de exemplo.

Create REST API

API details

☐ New API
Create a new REST API.

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

☐ Import API
Import an API from an OpenAPI definition.

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

```
1 {
2   "swagger": "2.0",
3   "info": {
4     "description": "Your first API with Amazon API Gateway. This is a sample API that integrates via HTTP w
ith our demo Pet Store endpoints",
5     "title": "PetStore"
6   },
7   "schemes": [
8     "https"
9   ],
10  "paths": {
11    "/": {
12      "get": {
13        "tags": [
14          "pets"
15        ],
16        "description": "PetStore HTML web page containing API usage information",
17        "consumes": [
18          "application/json"
19        ],
20      }
21    }
22  }
23 }
```

É possível percorrer a definição do OpenAPI para conhecer os detalhes dessa API de exemplo antes de selecionar Importar.

4. No painel de navegação principal, selecione Recursos. A API recém-criada é mostrada da seguinte forma:

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

Resources

API actions ▼

Deploy API

Create resource

☐ /

GET

☐ /pets

GET

OPTIONS

POST

☐ /{petId}

GET

OPTIONS

Resource details

Update documentation

Enable CORS

Path

/

Resource ID

efg456

Methods (1)

Delete

Create method

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

O painel Resources (Recursos) mostra a estrutura da API criada como uma árvore de nós. Os métodos de API definidos em cada recurso são as extremidades da árvore. Quando um recurso é selecionado, todos os respectivos métodos são listados na tabela Métodos à direita. Com cada método, são exibidos o tipo de método, o tipo de integração, o tipo de autorização e o requisito da chave de API.

5. Para visualizar os detalhes de um método, modificar sua configuração ou testar a invocação do método, escolha o nome do método na lista de métodos ou na árvore de recursos. Aqui, escolhemos o método POST /pets como uma ilustração:

Resources

Create resource

- /
- GET
- /pets
 - GET
 - OPTIONS
 - POST**
- /{petId}
 - GET
 - OPTIONS

/pets - POST - Method execution

ARN: `arn:aws:execute-api:us-west-1:111222333:abcd1234/*/POST/pets` Resource ID: `aaa111`

Update documentation Delete

Client → Method request → Integration request → HTTP integration → Integration response → Method response → Client

Method request Integration request Integration response Method response

Method request settings Edit

Authorization: NONE API key required: False

Request validator: None SDK operation name: CreatePet

Request paths (0)

Name	Caching

O painel Execução de método resultante apresenta uma visão lógica da estrutura e dos comportamentos do método escolhido (POST /pets).

A Solicitação de método e a Resposta de método representam a interface da API com o front-end, ao passo que a Solicitação de integração e a Resposta de integração representam a interface da API com o back-end.

Um cliente usa a API para acessar um recurso de back-end por meio da Solicitação de método. Se necessário, o API Gateway converte a solicitação do cliente no formato aceitável para o back-end em Solicitação de integração antes de encaminhá-la ao back-end. A solicitação transformada é conhecida como a solicitação de integração. De maneira semelhante, o back-end retorna a resposta ao API Gateway em Resposta de integração. Depois, o API Gateway a roteia para Method Response (Resposta de método) antes de enviá-la ao cliente. Novamente,

se necessário, o API Gateway poderá mapear os dados da resposta de backend para um formulário esperado pelo cliente.

Para o método POST em um recurso da API, a carga de solicitação de método poderá ser passada para a solicitação de integração sem modificação, se a carga de solicitação de método estiver no mesmo formato que a carga de solicitação de integração.

A solicitação do método GET / usa o tipo de integração MOCK e não está vinculada a nenhum endpoint de backend real. A Resposta de integração correspondente é configurada para gerar uma página HTML estática. Quando o método é chamado, o API Gateway simplesmente aceita a solicitação e retorna imediatamente a resposta de integração configurada para o cliente como Resposta de método. Você pode usar a integração fictícia para testar uma API sem exigir um endpoint de backend. Você também pode usá-la para atender a uma resposta local, gerada a partir de um modelo de mapeamento de corpo de resposta.

Como desenvolvedor de APIs, você pode controlar os comportamentos das interações de front-end da sua API, configurando a solicitação do método e uma resposta do método. Você controla os comportamentos das interações de backend da sua API configurando a solicitação de integração e a resposta da integração. Isso envolve mapeamentos de dados entre um método e sua integração correspondente. Por enquanto, nos concentraremos em testar a API para fornecer uma experiência de usuário de ponta a ponta.

6. Selecione a guia Testar. Talvez seja necessário selecionar o botão de seta para a direita para mostrar a guia.
7. Por exemplo, para testar o método POST /pets, digite a carga útil **{"type": "dog", "price": 249.99}** a seguir no Corpo da solicitação e escolha Testar.



Integration request

Integration response

Method response

Test

Test method

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

Query strings

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

Headers

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

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

Client certificate

No client certificates have been generated.

Request body

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

A entrada especifica os atributos do animal de estimação que queremos adicionar à lista de animais de estimação no site PetStore.

- Os resultados são exibidos da seguinte forma:

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

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

Response headers

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

Logs

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

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

A entrada Log da saída mostra as alterações de estado da solicitação de método para a solicitação de integração e da resposta de integração para a resposta de método. Isso pode ser útil para solucionar problemas com erros de mapeamento que causam a falha da solicitação. Neste exemplo, nenhum mapeamento é aplicado: a carga da solicitação do método é passada por meio de uma solicitação de integração para o backend e, de maneira semelhante, a resposta do backend é passada por meio da resposta da integração para a resposta do método.

Para testar a API usando um cliente diferente do recurso test-invoke-request do API Gateway, é necessário primeiro implantar a API em um estágio.

9. Para implantar o exemplo de API, selecione Implantar API.

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

Resources

API actions ▼

Deploy API

Create resource



/

GET



/pets

GET

OPTIONS

POST



/{petId}

GET

OPTIONS

Resource details

Path

/

Update documentation

Enable CORS

Resource ID

efg456

Methods (1)

Delete

Create method

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

10. Em Estágio, selecione Novo estágio e insira **test**.
11. (Opcional) Em Description (Descrição), insira uma descrição.
12. Escolha Implantar.
13. No painel Estágios resultante, em Detalhes do estágio, Invocar URL exibe o URL para invocar a solicitação de método GET / da API.

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

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

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

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

- Escolha o ícone de cópia para copiar o URL de invocação da API e insira o URL de invocação da sua API em um navegador da web. Uma resposta bem-sucedida retornará o resultado, gerada a partir do modelo de mapeamento na resposta de integração.
- No painel de navegação Estágios, expanda o estágio teste, selecione GET em / pets/{petId} e copie o valor de Invocar URL de `https://api-id.execute-api.region.amazonaws.com/test/pets/{petId}`. {petId} representa uma variável de caminho.

Cole o valor de Invoke URL (Invocar URL) (obtido na etapa anterior) na barra de endereços de um navegador, substituindo {petId} por, por exemplo, 1 e pressione Enter para enviar a solicitação. Uma resposta 200 OK será retornada com a seguinte carga JSON:

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

É possível invocar o método de API conforme mostrado porque seu tipo Authorization (Autorização) está definido como NONE. Se a autorização do AWS_IAM fosse usada, você assinaria a solicitação usando os protocolos [Signature Version 4](#) (SigV4) ou [Signature Version 4a](#) (SigV4a). Para obter um exemplo dessa solicitação, consulte [the section called “Tutorial: Crie uma API REST com uma integração de não proxy de HTTP”](#).

Escolher um tutorial de integração HTTP

Para criar uma API com integração HTTP, você pode usar uma integração de proxy HTTP ou uma integração HTTP personalizada.

Em uma integração de proxy HTTP, só é necessário definir o método HTTP e o URI de endpoint HTTP de acordo com os requisitos de back-end. Recomendamos o uso da integração de proxy HTTP sempre que possível para aproveitar a configuração simplificada da API.

Talvez você queira usar uma integração HTTP personalizada se precisar transformar os dados de solicitação do cliente para o back-end ou transformar os dados de resposta do back-end para o cliente.

Tópicos

- [Tutorial: Crie uma API REST com uma integração de proxy de HTTP](#)
- [Tutorial: Crie uma API REST com uma integração de não proxy de HTTP](#)

Tutorial: Crie uma API REST com uma integração de proxy de HTTP

A integração de proxy HTTP é um mecanismo simples, eficiente e versátil para criar uma API que permite que um aplicativo web acesse vários recursos ou características do endpoint HTTP integrado, por exemplo, todo o site, com a configuração simplificada de um único método da API. Na integração de proxy HTTP, o API Gateway transmite a solicitação de método enviada pelo cliente para o backend. Os dados da solicitação que são transmitidos incluem os cabeçalhos da solicitação, parâmetros da string de consulta, variáveis do caminho do URL e carga. O endpoint HTTP do backend ou o servidor web analisará os dados da solicitação recebida para determinar a resposta que ele retorna. A integração de proxy HTTP faz com que o cliente e o backend interajam diretamente sem qualquer intervenção do API Gateway após a configuração do método da API, exceto problemas conhecidos, como caracteres incompatíveis, que estão listados em [the section called “Observações importantes”](#).

Com o recurso de proxy totalmente abrangente {proxy+} e o verbo global ANY para o método HTTP, é possível usar uma integração de proxy HTTP para criar uma API de um único método de API. O método expõe o conjunto completo de recursos HTTP de acesso público e operações de um site. Quando o servidor web de backend abre mais recursos para o acesso público, o cliente pode usar esses novos recursos com a mesma configuração de API. Para habilitar isso, o desenvolvedor do site deve comunicar claramente ao desenvolvedor cliente quais são os novos recursos e quais operações são aplicáveis a cada um deles.

Como uma rápida introdução, o tutorial a seguir demonstra a integração de proxy HTTP. No tutorial, criamos uma API usando o console do API Gateway para integração com o site PetStore por meio de um recurso de proxy genérico {proxy+} e criamos o espaço reservado do método HTTP de ANY.

Tópicos

- [Criar uma API com integração de proxy HTTP usando o console do API Gateway](#)
- [Testar uma API com a integração de proxy HTTP](#)

Criar uma API com integração de proxy HTTP usando o console do API Gateway

O procedimento a seguir descreve as etapas para criar e testar uma API com um recurso de proxy para um backend HTTP usando o console do API Gateway. O backend HTTP é o site PetStore (<http://petstore-demo-endpoint.execute-api.com/petstore/pets>) de [Tutorial: Crie uma API REST com uma integração de não proxy de HTTP](#), no qual capturas de tela são usadas como auxílios visuais para ilustrar os elementos da IU do API Gateway. Se você ainda não sabe usar o console do API Gateway para criar uma API, talvez precise consultar essa seção primeiro.

Como criar uma API

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Se esta for a primeira vez que você usa o API Gateway, você verá uma página com os recursos do serviço. Em REST API, escolha Build (Criar). Quando o pop-up Create Example API (Criar API de exemplo) for exibido, escolha OK.

Se essa não for a primeira vez que você usa o API Gateway, escolha Create API (Criar API). Em REST API, escolha Build (Criar).

3. Em API name (Nome da API), insira **HTTPProxyAPI**.
4. (Opcional) Em Description (Descrição), insira uma descrição.
5. Mantenha Tipo de endpoint de API definido como Regional.
6. Em Tipo de endereço IP, selecione IPv4.
7. Selecione Criar API.

Nesta etapa, você vai criar um caminho de recurso de proxy de {proxy+}. Esse é o espaço reservado de qualquer um dos endpoints de back-end em <http://petstore-demo-endpoint.execute-api.com/>. Por exemplo, ele pode ser petstore, petstore/pets e

petstore/pets/{petId}. O API Gateway cria o método ANY quando você cria o recurso {proxy+} e serve como um espaço reservado para qualquer um dos verbos HTTP compatíveis em runtime.

Como criar um recurso /{proxy+}

1. Selecione a API.
2. No painel de navegação principal, selecione Recursos.
3. Selecione Criar recurso.
4. Ative Recurso proxy.
5. Mantenha Caminho do recurso como /.
6. Em Resource Name (Nome do recurso), insira **{proxy+}**.
7. Mantenha CORS (Compartilhamento de recursos de origem cruzada) desativado.
8. Selecione Criar recurso.

Create resource

Resource details

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

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

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

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

Nesta etapa, você vai integrar o método ANY a um endpoint HTTP de back-end usando uma integração de proxy. Na integração de proxy, o API Gateway transmite a solicitação de método enviada pelo cliente ao back-end, sem intervenção do API Gateway.

Como criar um método **ANY**

1. Selecione o recurso/{proxy+}.
2. Selecione o método ANY.
3. Sob o símbolo de aviso, selecione Editar integração. Não é possível implantar uma API que tenha um método sem uma integração.
4. Em Tipo de integração, selecione HTTP.
5. Ative Integração de proxy HTTP.
6. Em Método HTTP, selecione ANY.
7. Em URL do endpoint, insira **http://petstore-demo-endpoint.execute-api.com/{proxy}**.
8. Escolha Salvar.

Testar uma API com a integração de proxy HTTP

O êxito de determinada solicitação do cliente depende do seguinte:

- Se o backend disponibilizou o endpoint de backend correspondente e, se esse for o caso, se as permissões de acesso necessárias foram concedidas.
- Se o cliente forneceu a entrada correta.

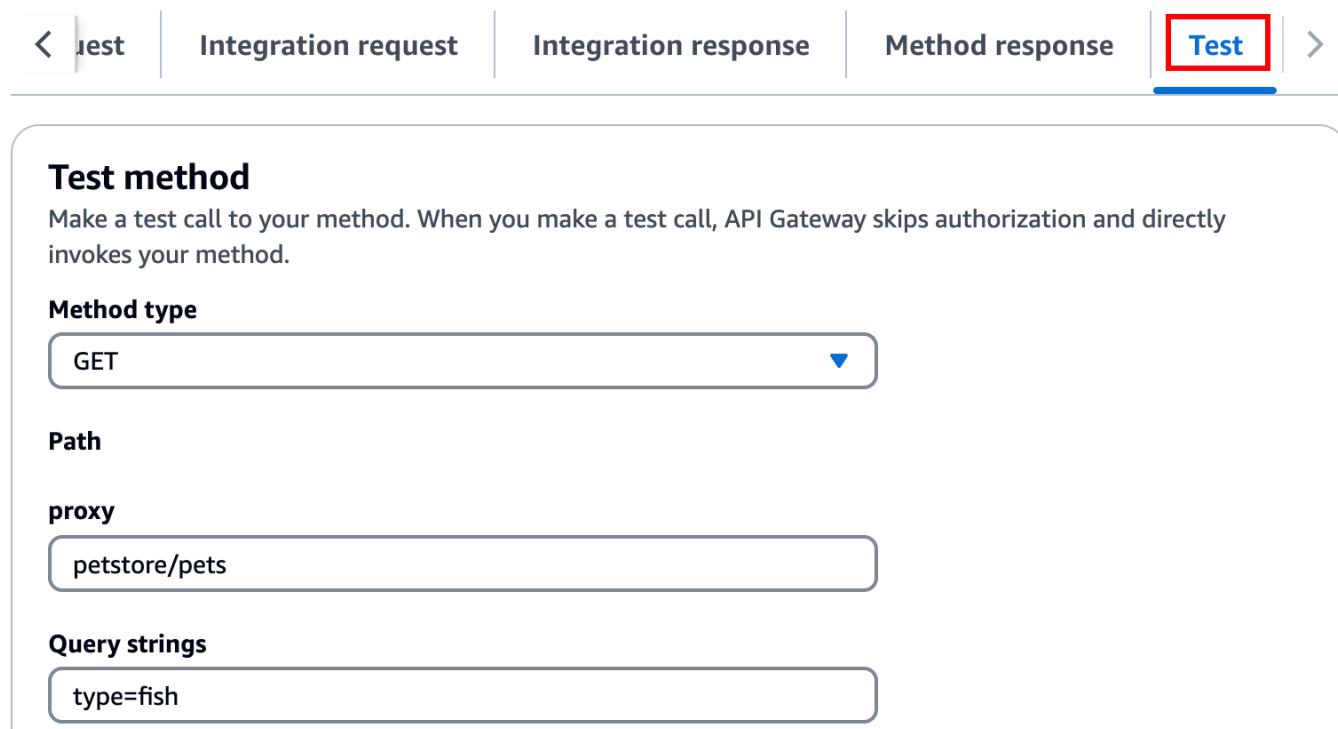
Por exemplo, a API da PetStore usada aqui não expõe o recurso /petstore. Dessa forma, você obtém uma resposta 404 Resource Not Found contendo a mensagem de erro Cannot GET /petstore.

Além disso, o cliente deve ser capaz de lidar com o formato de saída do backend para analisar o resultado corretamente. O API Gateway não se interpõe para facilitar as interações entre o cliente e o backend.

Para testar uma API integrada ao site PetStore usando a integração de proxy HTTP por meio do recurso de proxy

1. Selecione a guia Testar. Talvez seja necessário selecionar o botão de seta para a direita para mostrar a guia.
2. Em Tipo de método, selecione GET.
3. Em Caminho, em proxy, insira **petstore/pets**.

4. Em Strings de consulta, digite **type=fish**.
5. Escolha Testar.



The screenshot shows the 'Test' tab in the Amazon API Gateway console. The 'Method type' is set to GET. The 'Path' is /petstore/pets. The 'Query strings' are type=fish. The 'Test' button is highlighted with a red box.

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

Method type
GET

Path
/petstore/pets

Query strings
type=fish

Como o site de backend oferece suporte para a solicitação GET /petstore/pets? type=fish, ele retorna uma resposta bem-sucedida semelhante à seguinte:

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



```
]
```

Se você tentar chamar GET /petstore, receberá uma resposta 404 com uma mensagem de erro Cannot GET /petstore. Isso acontece porque o backend não oferece suporte à operação especificada. Se você chamar GET /petstore/pets/1, receberá uma resposta 200 OK com a carga a seguir, pois a solicitação recebe suporte do site PetStore.

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

Também é possível usar um navegador para testar a API. Implante a API e associe-a a um estágio para criar o URL de invocação da API.

Para implantar sua API

1. Escolha Implantar API.
2. Em Estágio, selecione Novo estágio.
3. Em Stage name (Nome do estágio), insira **test**.
4. (Opcional) Em Description (Descrição), insira uma descrição.
5. Escolha Implantar.

Agora, os clientes podem chamar sua API.

Como invocar a API

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione a API.
3. No painel de navegação principal, selecione Estágio.
4. Em Detalhes do estágio, escolha o ícone de cópia para copiar o URL de invocação da API.

Copie o URL de invocação da API em um navegador web.

O URL deve ser semelhante a `https://abcdef123.execute-api.us-east-2.amazonaws.com/test/petstore/pets?type=fish`.

Seu navegador envia uma solicitação GET à API.

5. O resultado deve ser o mesmo que o gerado ao usar Testar no console do API Gateway.

Tutorial: Crie uma API REST com uma integração de não proxy de HTTP

Neste tutorial, você criará uma API do zero usando o console do Amazon API Gateway. Você pode pensar no console como um estúdio de design de API e usá-lo para definir o escopo dos recursos da API, testar seus comportamentos, criar a API e implantá-la em estágios.

Tópicos

- [Criar uma API com integração personalizada HTTP](#)
- [\(Opcional\) Associar parâmetros de solicitação](#)

Criar uma API com integração personalizada HTTP

Esta seção orienta você pelas etapas necessárias para criar recursos, expor métodos em um recurso, configurar um método para alcançar os comportamentos de API desejados e testar e implantar a API.

Nesta etapa, você cria uma API vazia. Nas etapas a seguir, você vai criar recursos e métodos para conectar a API ao endpoint `http://petstore-demo-endpoint.execute-api.com/petstore/pets` usando uma integração HTTP sem proxy.

Como criar uma API

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Se esta for a primeira vez que você usa o API Gateway, você verá uma página com os recursos do serviço. Em REST API, escolha Build (Criar). Quando o pop-up Create Example API (Criar API de exemplo) for exibido, escolha OK.

Se essa não for a primeira vez que você usa o API Gateway, escolha Create API (Criar API). Em REST API, escolha Build (Criar).

3. Em API name (Nome da API), insira **HTTPNonProxyAPI**.
4. (Opcional) Em Description (Descrição), insira uma descrição.
5. Mantenha Tipo de endpoint de API definido como Regional.
6. Em Tipo de endereço IP, selecione IPv4.

7. Selecione Criar API.

A árvore Resources (Recursos) mostra o recurso raiz (/) sem métodos. Neste exercício, criaremos a API com a integração HTTP personalizada do site da PetStore (<http://petstore-demo-endpoint.execute-api.com/petstore/pets>.) Para fins de ilustração, vamos criar um recurso /pets como um filho da raiz e expor um método GET nesse recurso para um cliente recuperar uma lista de itens de animais de estimação disponíveis no site PetStore.

Como criar um recurso /pets

1. Selecione Criar recurso.
2. Mantenha Recurso proxy desativado.
3. Mantenha Caminho do recurso como /.
4. Em Resource Name (Nome do recurso), insira **pets**.
5. Mantenha CORS (Compartilhamento de recursos de origem cruzada) desativado.
6. Selecione Criar recurso.

Nesta etapa, você vai criar um método GET para o recurso /pets. O método GET é integrado ao site <http://petstore-demo-endpoint.execute-api.com/petstore/pets>. São outras opções para um método de API:

- POST, usado principalmente para criar recursos filho.
- PUT, usado principalmente para atualizar recursos existentes (e, embora não seja recomendado, pode ser usado para criar recursos filho).
- DELETE, usado para excluir recursos.
- PATCH, usado para atualizar recursos.
- HEAD, usado, principalmente em cenários de teste. É o mesmo que GET, mas não retorna a representação do recurso.
- OPTIONS, que pode ser usado por autores de chamadas para obter informações sobre opções de comunicação disponíveis para o serviço de destino.

Para o HTTP method (Método HTTP) da solicitação de integração, você deve escolher um que seja compatível com o backend. Para HTTP ou Mock integration, faz sentido que a solicitação de método e a solicitação de integração usem o mesmo verbo HTTP. Para outros tipos de integração,

a solicitação de método provavelmente usará um verbo HTTP diferente da solicitação de integração. Por exemplo, para chamar uma função do Lambda, a solicitação de integração deve usar POST para invocar a função, enquanto a solicitação de método pode usar qualquer verbo HTTP, dependendo da lógica da função do Lambda.

Como criar um método **GET** no recurso /pets

1. Selecione o recurso /pets.
2. Escolha Criar método.
3. Em Tipo de método, selecione GET.
4. Em Tipo de integração, selecione Integração HTTP.
5. Mantenha a opção Integração do proxy HTTP desativada.
6. Em Método HTTP, selecione GET.
7. Em URL do endpoint, insira **http://petstore-demo-endpoint.execute-api.com/petstore/pets**.

O site PetStore permite que você recupere uma lista de itens Pet por tipo de animal de estimação, por exemplo, “Cão” ou “Gato”, em uma página específica.

8. Em Manuseio de conteúdo, selecione Passagem.
9. Selecione Parâmetros de string de consulta de URL.

Ele usa os parâmetros de string de consulta `type` e `page` para aceitar uma entrada. Adicione os parâmetros de string de consulta à solicitação de método e mapeie-os para os parâmetros de string de consulta correspondentes da solicitação de integração.

10. Para adicionar parâmetros de string de consulta, faça o seguinte:
 - a. Escolha Add query string (Adicionar string de consulta).
 - b. Em Nome, insira **type**.
 - c. Mantenha Obrigatório e Armazenamento em cache desativados.

Repita as etapas anteriores para criar uma string de consulta adicional com o nome **page**.

11. Escolha Criar método.

O cliente agora pode fornecer um tipo de animal de estimação e um número de página como parâmetros de string de consulta ao enviar uma solicitação. Esses parâmetros de entrada devem ser

mapeados para parâmetros da sequência de consulta da integração para encaminhar os valores de entrada para nosso site PetStore no backend.

Como associar parâmetros de entrada à solicitação de integração

1. Na guia Solicitação de integração, em Configurações de solicitação de integração, selecione Editar.
2. Selecione Parâmetros de string de consulta de URL e faça o seguinte:
 - a. Selecione Adicionar parâmetro de string de consulta.
 - b. Em Nome, digite **type**.
 - c. Em Mapeado de, insira **method.request.querystring.type**.
 - d. Mantenha Armazenamento em cache desativado.
 - e. Selecione Adicionar parâmetro de string de consulta.
 - f. Em Nome, digite **page**.
 - g. Em Mapeado de, insira **method.request.querystring.page**.
 - h. Mantenha Armazenamento em cache desativado.
3. Escolha Salvar.

Como testar a API

1. Selecione a guia Testar. Talvez seja necessário selecionar o botão de seta para a direita para mostrar a guia.
2. Em Strings de consulta, digite **type=Dog&page=2**.
3. Escolha Testar.

O resultado é semelhante ao seguinte:

Method request

Integration request

Integration response

Method response

Test

Test method

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

Query strings

Headers

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

Client certificate

Test

/pets - GET method test results

Request

/pets?type=Dog&page=2

Latency ms

10

Status

200

Response body

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

Agora que o teste foi bem-sucedido, podemos implantar a API para torná-la disponível publicamente.

- Escolha Implantar API.
- Em Estágio, selecione Novo estágio.
- Em Stage name (Nome do estágio), insira **Prod**.
- (Opcional) Em Description (Descrição), insira uma descrição.
- Escolha Implantar.
- (Opcional) Em Detalhes do estágio, para Invocar URL, é possível selecionar o ícone de cópia para copiar o URL de invocação da API. Você pode usá-lo com ferramentas, como [Postman](#) e [cURL](#) para testar sua API.

Se você usar um SDK para criar um cliente, poderá chamar os métodos expostos pelo SDK para assinar a solicitação. Para obter detalhes da implementação, consulte o [SDK da AWS](#) de sua escolha.

 Note

Quando alterações são feitas na sua API, você deve reimplantá-la para disponibilizar os recursos novos ou atualizados antes de invocar novamente a URL da solicitação.

(Opcional) Associar parâmetros de solicitação

Mapear parâmetros de solicitação para uma API do API Gateway

Este tutorial mostra como criar um parâmetro de caminho de `{petId}` na solicitação de método da API para especificar um ID de item, associá-lo ao parâmetro de caminho `{id}` no URL da solicitação de integração e enviar a solicitação ao endpoint HTTP.

 Note

Se você digitar uma letra de tamanho incorreto, por exemplo, minúscula em vez de maiúscula, poderá causar erros mais adiante no processo.

Etapa 1: Criar recursos

Nesta etapa, você vai criar um recurso com um parâmetro de caminho `{petId}`.

Como criar o recurso `{petId}`

1. Selecione o recurso `/pets` e, depois, escolha Criar recurso.
2. Mantenha Recurso proxy desativado.
3. Em Caminho do recurso, selecione `/pets`.
4. Em Resource Name (Nome do recurso), insira **`{petId}`**.

Use as chaves (`{ }`) ao redor de `petId` para que `/pets/{petId}` seja exibido.

5. Mantenha CORS (Compartilhamento de recursos de origem cruzada) desativado.
6. Selecione Criar recurso.

Etapa 2: Criar e testar métodos

Nesta etapa, você vai criar um método GET com um parâmetro de caminho {petId}.

Como configurar o método GET

1. Selecione o recurso /{petId} e, depois, Criar método.
2. Em Tipo de método, selecione GET.
3. Em Tipo de integração, selecione Integração HTTP.
4. Mantenha a opção Integração do proxy HTTP desativada.
5. Em Método HTTP, selecione GET.
6. Em URL do endpoint, insira **http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}**.
7. Em Manuseio de conteúdo, selecione Passagem.
8. Mantenha o Tempo limite padrão ativado.
9. Escolha Criar método.

Agora você vai associar o parâmetro de caminho {petId} que você acabou de criar ao parâmetro de caminho {id} no URL do endpoint HTTP da solicitação de integração. O endpoint HTTP era **http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}**.

Como associar o parâmetro de caminho {petId}

1. Na guia Solicitação de integração, em Configurações de solicitação de integração, selecione Editar.
2. Selecione Parâmetros de caminho de URL.
3. O API Gateway cria um parâmetro de caminho para a solicitação de integração chamado petId, mas ele não é válido para o URL do endpoint HTTP que você define como a integração de backend. O endpoint HTTP usa {id} como o parâmetro de caminho. Em Nome, exclua petId e insira **id**.

Isso mapeia o parâmetro de caminho da solicitação de método de petId para o parâmetro de caminho da solicitação de integração de id.

4. Escolha Salvar.

Agora teste o método.

Como testar o método

1. Selecione a guia Testar. Talvez seja necessário selecionar o botão de seta para a direita para mostrar a guia.
2. Em Caminho para petId, insira **4**.
3. Escolha Test (Testar).

Se bem-sucedido, o Corpo da resposta exibirá o seguinte:

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

Etapa 3: implantar a API

Nesta etapa, você implantará a API para poder começar a chamá-la fora do console do API Gateway.

Para implantar a API

1. Escolha Implantar API.
2. Em Estágio, selecione Prod.
3. (Opcional) Em Description (Descrição), insira uma descrição.
4. Escolha Deploy (Implantar).

Etapa 4: testar a API

Nesta etapa, você sairá do console do API Gateway e usará sua API para acessar o endpoint HTTP.

1. No painel de navegação principal, selecione Estágio.
2. Em Detalhes do estágio, escolha o ícone de cópia para copiar o URL de invocação da API.

A aparência será semelhante à seguinte:

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

3. Insira este URL na caixa de endereço da nova guia do navegador e anexe `/pets/4` ao URL antes de enviar a solicitação.
4. O navegador exibirá o seguinte:

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

Próximas etapas

É possível personalizar ainda mais a API ativando a validação de solicitações, transformando dados ou criando respostas de gateway personalizadas.

Para examinar mais formas de personalizar a API, veja os seguintes tutoriais:

- Para obter mais informações sobre a validação de solicitação, consulte [Configurar a validação básica de solicitações no API Gateway](#).
- Para obter informações sobre como transformar cargas úteis de solicitação e resposta, consulte [Tutorial: modificar a solicitação e a resposta de integração para integrações a serviços da AWS](#).
- Para obter informações sobre como criar respostas de gateway personalizadas, consulte [Configurar uma resposta de gateway para uma API REST usando o console do API Gateway](#).

Tutorial: Crie uma API REST com uma integração privada

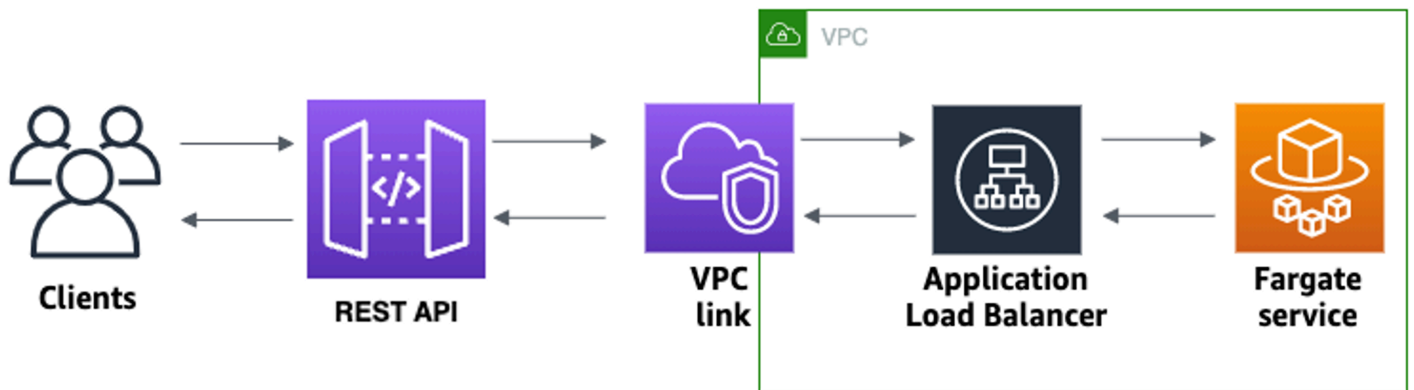
Neste tutorial, você vai criar uma API REST que se conecte a um serviço do Amazon ECS executado em uma Amazon VPC. Clientes fora da sua Amazon VPC podem usar a API para acessar seu serviço do Amazon ECS.

Este tutorial leva aproximadamente uma hora para ser concluído. Primeiro, você usa um modelo do CloudFormation para criar um serviço da Amazon VPC e do Amazon ECS. Depois, você vai usar o console do API Gateway para criar um link da VPC V2. O link da VPC permite que o API Gateway acesse o serviço do Amazon ECS executado em sua Amazon VPC. Depois, você vai criar uma API REST que use o link da VPC V2 para se conectar ao seu serviço do Amazon ECS. Por fim, você testa a sua API.

Quando você invoca sua API REST, o API Gateway encaminha a solicitação para o serviço do Amazon ECS por meio do link da VPC V2 e, depois, exibe a resposta do serviço.

Note

Anteriormente, esse tutorial comportava APIs HTTP e agora só comporta APIs REST que utilizem o link da VPC V2.



Para concluir esse tutorial, você precisa de uma conta da AWS e de um usuário do AWS Identity and Access Management com acesso ao console. Para obter mais informações, consulte [Pré-requisitos](#).

Tópicos

- [Etapa 1: Crie um serviço do Amazon ECS](#)
- [Etapa 2: Crie um link da VPC](#)
- [Etapa 3: criar uma API REST](#)
- [Etapa 4: Testar sua API](#)
- [Etapa 5: Implante sua API](#)
- [Etapa 6: chamar a API](#)
- [Etapa 7: limpar](#)

Etapa 1: Crie um serviço do Amazon ECS

O Amazon ECS é um serviço de gerenciamento de contêineres que facilita a execução, a interrupção e o gerenciamento de contêineres do Docker em um cluster. Neste tutorial, você executa seu cluster em uma infraestrutura sem servidor gerenciada pelo Amazon ECS.

Baixe e descompacte [este modelo do CloudFormation](#) que cria todas as dependências do serviço, incluindo uma Amazon VPC. Use o modelo para criar um serviço do Amazon ECS que usa um Application Load Balancer.

Como criar uma pilha do CloudFormation

1. Abra o console do CloudFormation em <https://console.aws.amazon.com/cloudformation>.
2. Selecione Create stack (Criar pilha) e With new resources (standard) (Com novos recursos (padrão)).
3. Em Specify template (Especificar modelo), escolha Upload a template file (Fazer upload de um arquivo de modelo).
4. Selecione o modelo que você baixou.
5. Escolha Next (Próximo).
6. Em Nome da pilha, insira **rest-api-private-integrations-tutorial** e escolha Avançar.
7. Para Configurar opções de pilha, escolha Avançar.
8. Para Capabilities (Recursos), reconheça que CloudFormation pode criar recursos do IAM em sua conta.
9. Escolha Próximo e, depois, Enviar.

CloudFormation provisiona o serviço ECS, que pode levar alguns minutos. Quando o status da sua pilha do CloudFormation for CREATE_COMPLETE, você estará pronto para passar para a próxima etapa.

Etapa 2: Crie um link da VPC

Um link da VPC permite que o API Gateway acesse recursos privados em uma Amazon VPC. Você deve usar um link da VPC para permitir que os clientes acessem seu serviço do Amazon ECS por meio da API REST.

Para criar um link da VPC

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, escolha Links da VPC e selecione Criar.

Escolha o ícone do menu para se for necessário abrir o painel de navegação principal.
3. Em Escolher versão do link de VPC, selecione Link da VPC V2.

4. Em Name (Nome), insira **private-integrations-tutorial**.
5. Em VPC, escolha a VPC criada na etapa 1. O nome deve começar com RestApiStack.
6. Para Sub-redes, selecione as duas sub-redes privadas em sua VPC. Os nomes delas terminam com PrivateSubnet.
7. Em Grupos de segurança, selecione o ID do grupo que começa com **private-integrations-tutorial** e tem a descrição de RestApiStack/RestApiTutorialService/Service/SecurityGroup.
8. Escolha Criar.

Depois de criar seu link da VPC V2, o API Gateway provisiona as interfaces de rede elásticas para acessar sua VPC. O processo pode levar alguns minutos. Enquanto isso, você pode criar sua API.

Etapa 3: criar uma API REST

A API REST fornece um endpoint HTTP para o seu serviço do Amazon ECS.

Como criar uma API REST

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha Criar API, em seguida, para API REST, escolha Criar.
3. Em Name (Nome), insira **private-integration-api**.
4. Em Tipo de endereço IP, selecione IPv4.
5. Selecione Criar API.

Depois de criar a API, você criará um método.

6. Selecione Criar método e proceda da seguinte maneira:
 - a. Em Tipo de método, selecione GET.
 - b. Em Tipo de integração, selecione Link de VPC.
 - c. Ative Integração de proxy VPC.
 - d. Em Método HTTP, selecione GET.
 - e. Em Link de VPC, escolha o link da VPC V2 que você criou na etapa anterior.
 - f. Em Destino de integração, insira o balanceador de carga criado com o modelo do CloudFormation na Etapa 1. Seu nome deve começar com rest-.
 - g. Em URL do endpoint, insira `http://private-integrations-tutorial.com`.

O URL é utilizado para definir o cabeçalho Host da solicitação de integração. Nesse caso, o cabeçalho do host é **private-integrations-tutorial**.

- h. Escolha Criar método.

Com a integração de proxy, a API está pronta para o teste.

Etapa 4: Testar sua API

Depois, teste a invocação do método de API.

Para testar sua API

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione a API.
3. Selecione a guia Testar. Talvez seja necessário selecionar o botão de seta para a direita para mostrar a guia.
4. Escolha Testar.

Verifique se a resposta da sua API é uma mensagem de boas-vindas que informa que seu aplicativo está sendo executado no Amazon ECS.

Etapa 5: Implante sua API

Depois, implante a API.

Para implantar sua API

1. Escolha Implantar API.
2. Em Estágio, selecione Novo estágio.
3. Em Stage name (Nome do estágio), insira **Prod**.
4. (Opcional) Em Description (Descrição), insira uma descrição.
5. Escolha Implantar.

Etapa 6: chamar a API

Depois de implantar a API, você pode chamá-la.

Como chamar a API

1. Insira o URL de invocação em um navegador da web.

O URL deve ser semelhante a `https://abcd123.execute-api.us-east-2.amazonaws.com/Prod`.

Seu navegador envia uma solicitação GET à API.

2. Verifique se a resposta da sua API é uma mensagem de boas-vindas que informa que seu aplicativo está sendo executado no Amazon ECS.

Se você vir a mensagem de boas-vindas, é sinal de que criou corretamente um serviço do Amazon ECS executado em uma Amazon VPC e utilizou uma API REST do API Gateway com um link da VPC V2 para acessar o serviço do Amazon ECS.

Etapa 7: limpar

Para evitar custos desnecessários, exclua os recursos que você criou como parte desse tutorial. As etapas a seguir excluem o link da VPC V2, a pilha do CloudFormation e a API REST.

Para excluir uma API REST

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Na página APIs , selecione uma API. Escolha Ações, escolha Excluir e, em seguida, confirme sua escolha.

Para excluir um link da VPC

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha Link da VPC.
3. Selecione seu link da VPC, escolha Excluir e confirme sua escolha.

Para excluir uma pilha do CloudFormation

1. Abra o console do CloudFormation em <https://console.aws.amazon.com/cloudformation>.
2. Selecione sua pilha do CloudFormation.
3. Escolha Excluir e, em seguida, confirme sua escolha.

Tutorial: Crie uma API REST com uma integração da AWS

Os tópicos [Tutorial: Crie uma API REST com uma integração de proxy do Lambda](#) e [Tutorial: Crie uma API REST com uma integração de não proxy do Lambda](#) descrevem como criar uma API do API Gateway para expor a função do Lambda integrada. Além disso, é possível criar uma API do API Gateway para expor outros serviços da AWS, como Amazon SNS, Amazon S3, Amazon Kinesis e até mesmo o AWS Lambda. Isso é possível através da integração da AWS. A integração do Lambda ou a integração de proxy do Lambda é um caso especial, em que a invocação da função do Lambda é exposta por meio da API do API Gateway.

Todos os serviços da AWS são compatíveis com APIs dedicadas para exposição de seus recursos. No entanto, os protocolos de aplicativos ou interfaces de programação provavelmente diferem de serviço para serviço. Uma API do API Gateway com integração da AWS tem a vantagem de fornecer um protocolo de aplicação consistente para que seu cliente acesse diferentes serviços da AWS.

Nesta demonstração, criamos uma API para expor o Amazon SNS. Para obter mais exemplos de integração de uma API a outros serviços da AWS, consulte [Tutoriais e workshops do Amazon API Gateway](#).

Diferentemente da integração de proxy do Lambda não há integração de proxy correspondente para outros serviços da AWS. Portanto, um método de API é integrado a uma única ação da AWS. Para obter mais flexibilidade, semelhante à integração de proxy, é possível configurar uma integração de proxy do Lambda. A função do Lambda então analisa e processa as solicitações para outras ações da AWS.

O API Gateway não tenta novamente quando o endpoint atinge o tempo limite. O autor da chamada da API deve implementar lógica de novas tentativas para lidar com tempos limite do endpoint.

Este passo a passo se baseia nas instruções e nos conceitos em [Tutorial: Crie uma API REST com uma integração de não proxy do Lambda](#). Se você ainda não concluiu esse passo-a-passo, sugerimos que faça isso primeiro.

Tópicos

- [Pré-requisitos](#)
- [Etapa 1: Criar o perfil de execução do proxy de serviço da AWS](#)
- [Etapa 2: Criar o recurso](#)
- [Etapa 3: Criar o método GET](#)
- [Etapa 4: Especificar configurações de método e testar o método](#)

- [Etapa 5: implantar a API](#)
- [Etapa 6: testar a API](#)
- [Etapa 7: Limpeza](#)

Pré-requisitos

Antes de iniciar esta demonstração, você deve fazer o seguinte:

1. Siga as etapas em [Configurar para usar o API Gateway](#).
2. Criar uma nova API chamada MyDemoAPI. Para obter mais informações, consulte [Tutorial: Crie uma API REST com uma integração de não proxy de HTTP](#).
3. Implante a API pelo menos uma vez em um estágio denominado test. Para obter mais informações, consulte [Implantar a API](#) em [Escolher um tutorial de integração do AWS Lambda](#).
4. Conclua as demais etapas em [Escolher um tutorial de integração do AWS Lambda](#).
5. Crie pelo menos um tópico no Amazon Simple Notification Service (Amazon SNS). Você usará a API implantada para obter uma lista de tópicos no Amazon SNS que estão associados à sua conta da AWS. Para saber como criar um tópico no Amazon SNS, consulte [Criar um tópico](#). (Não é necessário copiar o ARN do tópico mencionado na etapa 5.)

Etapa 1: Criar o perfil de execução do proxy de serviço da AWS

Para que a API invoque ações do Amazon SNS, é necessário que as políticas do IAM apropriadas sejam associadas a um perfil do IAM. Neste exercício, você cria um perfil do IAM.

Como criar o perfil de execução do proxy de serviço da AWS

1. Faça login no Console de gerenciamento da AWS e abra o console do IAM em <https://console.aws.amazon.com/iam/>.
2. Escolha Perfis.
3. Selecione Criar perfil.
4. Selecione Serviço da AWS em Selecionar tipo de entidade confiável, selecione API Gateway e Permite que o API Gateway envie logs ao CloudWatch Logs.
5. Selecione Próximo e, depois, Próximo.
6. Em Role name (Nome da função), digite **APIGatewaySNSProxyPolicy** e escolha Create role (Criar função).

7. Na lista Roles (Funções), escolha a função que você acaba de criar. Talvez seja necessário rolar a página ou usar a barra de pesquisa para encontrar o perfil.
8. Para a função escolhida, selecione a guia Adicionar permissões.
9. Selecione Anexar políticas na lista suspensa.
10. Na barra de pesquisa, insira **AmazonSNSReadOnlyAccess** e selecione Adicionar permissões.

 Note

Este tutorial usa uma política gerenciada em prol da simplicidade. Como prática recomendada, você deve criar sua própria política do IAM para conceder as permissões mínimas necessárias.

11. Anote o ARN do perfil recém-criado, você o usará posteriormente.

Etapa 2: Criar o recurso

Nesta etapa, crie um recurso que permite que o proxy de serviço da AWS interaja com o serviço da AWS.

Para criar o recurso

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione a API.
3. Selecione o recurso raiz, /, representado por uma única barra (/) e, depois, selecione Criar recurso.
4. Mantenha Recurso proxy desativado.
5. Mantenha Caminho do recurso como /.
6. Em Resource Name (Nome do recurso), insira **mydemoawsproxy**.
7. Mantenha CORS (Compartilhamento de recursos de origem cruzada) desativado.
8. Selecione Criar recurso.

Etapa 3: Criar o método GET

Nesta etapa, crie um método GET que permite que o proxy de serviço da AWS interaja com o serviço da AWS.

Como criar o método **GET**

1. Selecione o recurso `/mydemoawsproxy` e Criar método.
2. Em tipo de método, selecione GET.
3. Em Tipo de integração, selecione AWS service (Serviço da AWS).
4. Em Região da AWS, selecione a Região da AWS onde você criou o tópico do Amazon SNS.
5. Para AWS service (Serviço da AWS), selecione Amazon SNS.
6. Mantenha o subdomínio da AWS em branco.
7. Em Método HTTP, selecione GET.
8. Em Tipo de ação, selecione Usar nome da ação.
9. Em Nome da ação, insira **ListTopics**.
10. Em Perfil de execução, digite o ARN do perfil para **APIGatewaySNSProxyPolicy**.
11. Escolha Criar método.

Etapa 4: Especificar configurações de método e testar o método

Agora, é possível testar o método GET para verificar se ele foi configurado corretamente para listar os tópicos do Amazon SNS.

Como testar o método **GET**

1. Selecione a guia Testar. Talvez seja necessário selecionar o botão de seta para a direita para mostrar a guia.
2. Escolha Testar.

O resultado exibe uma resposta semelhante ao seguinte:

```
{
  "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"
}
}
```

Etapa 5: implantar a API

Nesta etapa, você implantará a API para poder chamá-la de fora do console do API Gateway.

Para implantar a API

1. Escolha Implantar API.
2. Em Estágio, selecione Novo estágio.
3. Em Stage name (Nome do estágio), insira **test**.
4. (Opcional) Em Description (Descrição), insira uma descrição.
5. Escolha Deploy (Implantar).

Etapa 6: testar a API

Nesta etapa, você sairá do console do API Gateway e usará seu proxy de serviço da AWS para interagir com o serviço Amazon SNS.

1. No painel de navegação principal, selecione Estágio.
2. Em Detalhes do estágio, escolha o ícone de cópia para copiar o URL de invocação da API.

A aparência deve ser semelhante a esta:

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

3. Cole o URL na caixa de endereço de uma nova guia do navegador.
4. Acrescente /mydemoawsproxy para que o URL tenha a seguinte aparência:

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

Navegue até a URL. Devem ser exibidas informações semelhante às seguintes:

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

Etapa 7: Limpeza

É possível excluir os recursos da IAM necessários que o proxy de serviço da AWS precisa para trabalhar.

Warning

Se você excluir um recurso do IAM do qual o proxy de serviço da AWS depende, esse proxy de serviço da AWS e qualquer API que depender dele deixarão de funcionar. A exclusão de um recurso do IAM não pode ser desfeita. Se quiser usar o recurso do IAM novamente, será necessário recriá-lo.

Como excluir os recursos do IAM associados

1. Abra o console do IAM em <https://console.aws.amazon.com/iam/>.
2. Na área Details (Detalhes), escolha Roles (Funções).
3. Selecione APIGatewayAWSProxyExecRole e, depois, escolha Role Actions (Ações da função), Delete Role (Excluir função). Quando solicitado, escolha Yes, Delete (Sim, excluir).
4. Na área Details (Detalhes), escolha Policies (Políticas).
5. Selecione APIGatewayAWSProxyExecPolicy e, depois, escolha Policy Actions (Ações da política), Delete (Excluir). Quando solicitado, escolha Delete (Excluir).

Você chegou ao final deste passo a passo. Para discussões mais detalhadas sobre como criar uma API como um proxy de serviço da AWS, consulte [Tutorial: Crie uma API REST como um proxy do Amazon S3](#), [Tutorial: Criar uma API REST de calculadora com duas integrações de serviços da](#)

[AWS e uma integração sem proxy do Lambda](#), ou [Tutorial: Criar uma API REST como um proxy do Amazon Kinesis](#).

Tutorial: Criar uma API REST de calculadora com duas integrações de serviços da AWS e uma integração sem proxy do Lambda

O [the section called “Tutorial: Crie uma API REST com uma integração de não proxy do Lambda”](#) usa a integração Lambda Function exclusivamente. A integração Lambda Function é um caso especial do tipo de integração do AWS Service que executa grande parte da configuração de integração para você, como adicionar automaticamente as permissões necessárias com base em recursos para invocar a função do Lambda. Aqui, duas das três integrações usam a integração AWS Service. Nesse tipo de integração, você tem mais controle, mas precisará executar tarefas manualmente, como criar e especificar uma função do IAM que contém as permissões adequadas.

Neste tutorial, você criará uma função do Lambda Calc que implementa operações aritméticas básicas, aceitando e retornando a entrada e a saída formatadas em JSON. Você vai criar uma API REST e integrá-la à função do Lambda das seguintes formas:

1. Ao expor um método GET no recurso `/calc` para invocar a função do Lambda, fornecendo a entrada como parâmetros de string de consulta. (Integração AWS Service)
2. Ao expor um método POST no recurso `/calc` para invocar a função do Lambda, fornecendo a entrada na carga de solicitação do método. (Integração AWS Service)
3. Ao expor um GET em recursos `/calc/{operand1}/{operand2}/{operator}` aninhados para invocar a função do Lambda, fornecendo a entrada como parâmetros de caminho. (Integração Lambda Function)

Além de experimentar este tutorial, talvez você queira estudar o [arquivo de definição do OpenAPI](#) para a API Calc, que pode ser importado para o API Gateway seguindo as instruções em [the section called “OpenAPI”](#).

Tópicos

- [Criar uma função do IAM que pode ser assumida](#)
- [Criar uma função do Lambda Calc](#)
- [Testar a função do Lambda Calc](#)
- [Criar uma API Calc](#)

- [Integração 1: Criar um método GET com parâmetros de consulta para chamar a função do Lambda](#)
- [Integração 2: Criar um método POST com uma carga JSON para chamar a função do Lambda](#)
- [Integração 3: Criar um método GET com parâmetros de caminho para chamar a função do Lambda](#)
- [Definições do OpenAPI da API demonstrativa integrada com uma função do Lambda](#)

Criar uma função do IAM que pode ser assumida

Para que sua API invoque sua função do Lambda Calc, você precisará ter uma função do IAM do API Gateway que pode ser assumida, que é uma função do IAM com o seguinte relacionamento confiável:

JSON

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

A função criada precisará ter uma permissão do Lambda [InvokeFunction](#). Caso contrário, o autor da chamada de API receberá uma resposta 500 Internal Server Error. Para conceder essa permissão à função, você anexará a política do IAM a ela:

JSON

```
{
  "Version": "2012-10-17",
```

```
    "Statement": [  
      {  
        "Effect": "Allow",  
        "Action": "lambda:InvokeFunction",  
        "Resource": "*"   
      }  
    ]  
  }  
}
```

Veja a seguir como fazer isso:

Criar uma função do IAM do API Gateway que possa ser assumida

1. Faça login no console do IAM.
2. Escolha Roles.
3. Selecione Create Role.
4. Em Select type of trusted entity (Selecionar tipo de entidade confiável), selecione AWS Serviço.
5. Em Choose the service that will use this role (Escolha o serviço que usará essa função), escolha Lambda.
6. Escolha Next: Permissions (Próximo: Permissões).
7. Selecione Create Policy (Criar política).

Uma nova janela do console Create Policy (Criar Política) será aberta. Nessa janela, faça o seguinte:

- a. Na guia JSON, substitua a política existente pela política a seguir:

JSON

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



```
]
}
```

- b. Escolha Review policy (Revisar política).
 - c. Em Review Policy (Revisar política), faça o seguinte:
 - i. Em Name, digite um nome de usuário, como **lambda_execute**.
 - ii. Selecione Create Policy (Criar política).
8. Na janela do console Create Role (Criar função) original, faça o seguinte:
 - a. Em Attach permissions policies (Anexar permissões políticas), escolha a política **lambda_execute** na lista suspensa.

Se você não vir a política na lista, selecione o botão Atualizar na parte superior da lista. (Não atualize a página do navegador.)
 - b. Selecione Next: Tags (Próximo: tags).
 - c. Selecione Next:Review (Próximo: análise).
 - d. Em Role name (Nome da função), digite um nome como **lambda_invoke_function_assume_apigw_role**.
 - e. Selecione Create role.
9. Escolha **lambda_invoke_function_assume_apigw_role** na lista de funções.
10. Selecione a guia Trust relationships (Relações de confiança).
11. Escolha Edit trust relationship (Editar relação de confiança).
12. Substitua a política existente pela seguinte:

JSON

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

```
        "lambda.amazonaws.com",
        "apigateway.amazonaws.com"
    ],
    },
    "Action": "sts:AssumeRole"
}
]
```

13. Escolha Update Trust Policy.

14. Anote o ARN da função para a função que acabou de criar. Você precisará disso mais tarde.

Criar uma função do Lambda **Calc**

Depois, você criará uma função do Lambda usando o console do Lambda.

1. No console do Lambda, escolha Create function (Criar função).
2. Escolha Author from Scratch.
3. Em Name (Nome), insira **Calc**.
4. Em Tempo de execução, escolha o último runtime Node.js ou Python compatível.
5. Em todas as outras opções, use a configuração padrão.
6. Escolha a opção Criar função.
7. Copie a função do Lambda a seguir em seu runtime preferido e cole-a no editor de código do console do 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. Em Execution role (Função de execução), selecione Choose an existing role (Selecionar uma função existente).
9. Insira o ARN da função para a função **lambda_invoke_function_assume_apigw_role** que você criou anteriormente.
10. Escolha Deploy (Implantar).

Essa função requer dois operandos (a e b) e um operador (op) a partir do parâmetro de entrada event. A entrada é um objeto JSON no seguinte formato:

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

```
"op": "String"
}
```

Essa função retorna o resultado calculado (c) e a entrada. Para uma entrada inválida, a função retorna o valor nulo ou a string "Invalid op" como resultado. A saída é do seguinte formato JSON:

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

É necessário testar a função no console do Lambda antes de integrá-la à API na próxima etapa.

Testar a função do Lambda **Calc**

Veja a seguir como testar a função **Calc** no console do Lambda:

1. Selecione a guia Testar.
2. Para o nome de evento de teste, insira **calc2plus5**.
3. Substitua a definição de eventos de teste pelo seguinte:

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

4. Escolha Save (Salvar).
5. Escolha Test (Testar).
6. Expanda Execution result: succeeded (Resultado da execução: com êxito). Você deve ver o seguinte:

```
{
  "a": 2,
  "b": 5,
```

```
"op": "+",  
"c": 7  
}
```

Criar uma API **Calc**

O procedimento a seguir mostra como criar uma API para a função do Lambda **Calc** que você acabou de criar. Nas seções a seguir, você adicionará recursos e métodos a ela.

Como criar uma API

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Se esta for a primeira vez que você usa o API Gateway, você verá uma página com os recursos do serviço. Em REST API, escolha Build (Criar). Quando o pop-up Create Example API (Criar API de exemplo) for exibido, escolha OK.

Se essa não for a primeira vez que você usa o API Gateway, escolha Create API (Criar API). Em REST API, escolha Build (Criar).

3. Em API name (Nome da API), insira **LambdaCalc**.
4. (Opcional) Em Description (Descrição), insira uma descrição.
5. Mantenha Tipo de endpoint de API definido como Regional.
6. Em Tipo de endereço IP, selecione IPv4.
7. Selecione Create API (Criar API).

Integração 1: Criar um método **GET** com parâmetros de consulta para chamar a função do Lambda

Ao criar um método GET que transmita parâmetros de string de consulta para a função do Lambda, habilite a API para ser invocada de um navegador. Essa abordagem pode ser útil, especialmente para APIs que permitem acesso aberto.

Depois de criar uma API, você criará um recurso. Normalmente, os recursos da API são organizados em uma árvore de recursos, de acordo com a lógica do aplicativo. Para esta etapa, você vai criar um recurso `/calc`.

Como criar um recurso /calc

1. Selecione Criar recurso.
2. Mantenha Recurso proxy desativado.
3. Mantenha Caminho do recurso como /.
4. Em Resource Name (Nome do recurso), insira **calc**.
5. Mantenha CORS (Compartilhamento de recursos de origem cruzada) desativado.
6. Selecione Criar recurso.

Ao criar um método GET que transmita parâmetros de string de consulta para a função do Lambda, habilite a API para ser invocada de um navegador. Essa abordagem pode ser útil, especialmente para APIs que permitem acesso aberto.

Neste método, o Lambda requer que a solicitação POST seja usada para invocar qualquer função do Lambda. Esse exemplo mostra que o método HTTP em uma solicitação de método de front-end pode ser diferente da solicitação de integração no backend.

Como criar um método GET

1. Selecione o recurso /calc e Criar método.
2. Em Tipo de método, selecione GET.
3. Em Tipo de integração, selecione AWS service (Serviço da AWS).
4. Em Região da AWS, selecione a Região da AWS onde você criou a função do Lambda.
5. Em AWS service (Serviço da AWS), selecione Lambda.
6. Mantenha o subdomínio da AWS em branco.
7. Em Método HTTP, selecione POST.
8. Em Tipo de ação, escolha Usar substituição de caminho. Essa opção nos permite especificar o ARN da ação [Invoke](#) para executar nossa função Calc.
9. Em Substituição de caminho, digite **2015-03-31/functions/arn:aws:lambda:us-east-2:account-id:function:Calc/invocations**. Em **account-id**, insira o ID da Conta da AWS. Em **us-east-2**, insira a Região da AWS onde você criou a função do Lambda.
10. Em Perfil de execução, digite o ARN do perfil para **lambda_invoke_function_assume_apigw_role**.
11. Não altere as configurações do Cache de credenciais e do Tempo limite padrão.

12. Escolha Configurações de solicitação de método.
13. Em Validador de solicitação, selecione Validar parâmetros de string de consulta e cabeçalhos.

Essa configuração fará com que uma mensagem de erro seja gerada se o cliente não especificar os parâmetros necessários.

14. Selecione Parâmetros de string de consulta de URL.

Agora configure parâmetros de string de consulta para o método GET no recurso /calc para que ele possa receber a entrada em nome da função do Lambda de back-end.

Para criar parâmetros de string de consulta, faça o seguinte:

- a. Escolha Add query string (Adicionar string de consulta).
- b. Em Nome, digite **operand1**.
- c. Ative a opção Obrigatório.
- d. Mantenha Armazenamento em cache desativado.

Repita as mesmas etapas e crie uma string de consulta chamada **operand2** e uma string de consulta chamada **operator**.

15. Escolha Criar método.

Agora, você vai criar um modelo de mapeamento para converter as strings de consulta fornecidas pelo cliente na carga útil de solicitações de integração, conforme exigido pela função `Calc`. Esse modelo associa os três parâmetros de consulta declarados em Solicitação de método a valores de propriedade designados do objeto JSON como entrada para a função do Lambda de back-end. O objeto JSON transformado será incluído como a carga útil da solicitação de integração.

Como associar parâmetros de entrada à solicitação de integração

1. Na guia Solicitação de integração, em Configurações de solicitação de integração, selecione Editar.
2. Em Passagem do corpo da solicitação, selecione Quando não há modelos definidos (recomendado).
3. Selecione Modelos de mapeamento.
4. Escolha Add mapping template (Adicionar modelo de mapeamento).
5. Em Tipo de conteúdo, insira **application/json**.

6. Em Corpo do modelo, insira o seguinte código:

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

7. Escolha Salvar.

Agora, é possível testar o método GET para verificar se ele foi configurado corretamente para invocar a função do Lambda:

Como testar o método **GET**

1. Selecione a guia Testar. Talvez seja necessário selecionar o botão de seta para a direita para mostrar a guia.
2. Em Strings de consulta, digite **operand1=2&operand2=3&operator=+**.
3. Escolha Test (Testar).

Os resultados devem ser semelhantes ao seguinte:

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

Integração 2: Criar um método **POST** com uma carga JSON para chamar a função do Lambda

Depois da criação de um método POST com uma carga JSON para chamar a função do Lambda, o cliente deve enviar a entrada necessária para a função de backend no corpo da solicitação. Para garantir que o cliente faça upload dos dados de entrada corretos, você habilitará a validação de solicitações na carga.

Como criar um método **POST** com uma carga útil do JSON

1. Selecione o recurso `/calc` e Criar método.
2. Em Tipo de método, selecione POST.
3. Em Tipo de integração, selecione AWS service (Serviço da AWS).
4. Em Região da AWS, selecione a Região da AWS onde você criou a função do Lambda.
5. Em AWS service (Serviço da AWS), selecione Lambda.
6. Mantenha o subdomínio da AWS em branco.
7. Em Método HTTP, selecione POST.
8. Em Tipo de ação, escolha Usar substituição de caminho. Essa opção nos permite especificar o ARN da ação [Invoke](#) para executar nossa função `Calc`.
9. Em Substituição de caminho, digite **2015-03-31/functions/arn:aws:lambda:us-east-2:account-id:function:Calc/invocations**. Em **account-id**, insira o ID da Conta da AWS. Em **us-east-2**, insira a Região da AWS onde você criou a função do Lambda.
10. Em Perfil de execução, digite o ARN do perfil para **lambda_invoke_function_assume_apigw_role**.
11. Não altere as configurações do Cache de credenciais e do Tempo limite padrão.
12. Escolha Criar método.

Agora você vai criar um modelo de entrada para descrever a estrutura de dados de entrada e validar o corpo da solicitação recebida.

Como criar o modelo de entrada

1. No painel de navegação principal, selecione Modelos.
2. Escolha Criar modelo.
3. Em Nome, digite **input**.
4. Em Tipo de conteúdo, insira **application/json**.

Se nenhum tipo de conteúdo correspondente for encontrado, a validação da solicitação não será executada. Para usar o mesmo modelo, independentemente do tipo de conteúdo, insira **\$default**.

5. Em Esquema do modelo, insira o seguinte modelo:

```
{
```

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

6. Escolha Criar modelo.

Agora você vai criar um modelo de saída. Esse modelo descreve a estrutura de dados da saída calculada do backend. Ele pode ser usado para mapear dados de resposta de integração para um modelo diferente. Este tutorial depende do comportamento de passagem direta e não usa este modelo.

Como criar um modelo de saída

1. Escolha Criar modelo.
2. Em Nome, digite **output**.
3. Em Tipo de conteúdo, insira **application/json**.

Se nenhum tipo de conteúdo correspondente for encontrado, a validação da solicitação não será executada. Para usar o mesmo modelo, independentemente do tipo de conteúdo, insira **\$default**.

4. Em Esquema do modelo, insira o seguinte modelo:

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

5. Escolha Criar modelo.

Agora você vai criar um modelo de resultado. Esse modelo descreve a estrutura de dados dos dados de resposta retornados. Ele faz referência aos esquemas de entrada e de saída definidos na API.

Como criar um modelo de resultado

1. Escolha Criar modelo.
2. Em Nome, digite **result**.
3. Em Tipo de conteúdo, insira **application/json**.

Se nenhum tipo de conteúdo correspondente for encontrado, a validação da solicitação não será executada. Para usar o mesmo modelo, independentemente do tipo de conteúdo, insira **\$default**.

4. Em Esquema do modelo, insira o modelo a seguir com o *restapi-id*. O *restapi-id* está listado entre parênteses na parte superior do console no seguinte fluxo: 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. Escolha Criar modelo.

Agora, você vai configurar a solicitação do método POST para habilitar a validação de solicitações no corpo da solicitação recebida.

Habilitar a validação da solicitação no método POST

1. No painel de navegação principal, selecione Recursos e, depois, selecione o método POST na árvore de recursos.
2. Na guia Solicitação de método, em Configurações de solicitação de método, escolha Editar.
3. Em Validador de solicitação, selecione Validar corpo.

4. Selecione Corpo da solicitação e, depois, Adicionar modelo.
5. Em Tipo de conteúdo, insira **application/json**.

Se nenhum tipo de conteúdo correspondente for encontrado, a validação da solicitação não será executada. Para usar o mesmo modelo, independentemente do tipo de conteúdo, insira **\$default**.

6. Em Modelo, selecione entrada.
7. Escolha Salvar.

Agora, é possível testar o método POST para verificar se ele foi configurado corretamente para invocar a função do Lambda:

Como testar o método **POST**

1. Selecione a guia Testar. Talvez seja necessário selecionar o botão de seta para a direita para mostrar a guia.
2. Em Corpo da solicitação, insira a carga útil do JSON a seguir.

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

3. Escolha Testar.

A seguinte saída deverá ser mostrada:

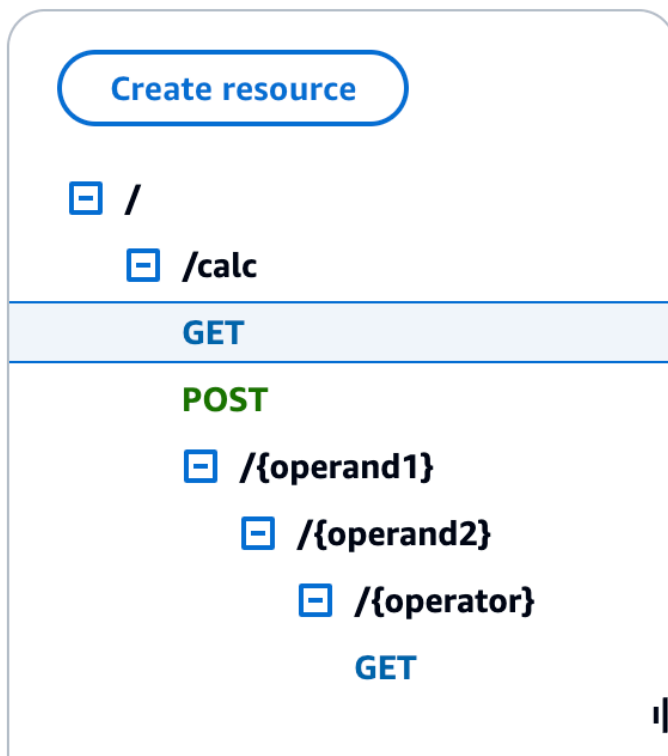
```
{
  "a": 1,
  "b": 2,
  "op": "+",
  "c": 3
}
```

Integração 3: Criar um método **GET** com parâmetros de caminho para chamar a função do Lambda

Agora, você criará um método GET em um recurso especificado por uma sequência de parâmetros de caminho para chamar a função do Lambda de backend. Os valores de parâmetros de caminho especificam os dados de entrada para a função do Lambda. Você usará um modelo de mapeamento para mapear os valores de parâmetro de caminho de entrada para a carga necessária de solicitação de integração.

A aparência da estrutura de recursos de API resultante será semelhante a esta:

Resources



Como criar um recurso `/operand1/operand2/operator`

1. Selecione Criar recurso.
2. Em Caminho do recurso, selecione `/calc`.
3. Em Resource Name (Nome do recurso), insira **`{operand1}`**.
4. Mantenha CORS (Compartilhamento de recursos de origem cruzada) desativado.
5. Selecione Criar recurso.

6. Em Caminho do recurso, selecione `/calc/{operand1}/`.
7. Em Resource Name (Nome do recurso), insira **{operand2}**.
8. Mantenha CORS (Compartilhamento de recursos de origem cruzada) desativado.
9. Selecione Criar recurso.
10. Em Caminho do recurso, selecione `/calc/{operand1}/{operand2}/`.
11. Em Resource Name (Nome do recurso), insira **{operator}**.
12. Mantenha CORS (Compartilhamento de recursos de origem cruzada) desativado.
13. Selecione Criar recurso.

Dessa vez, você usará a integração do Lambda incorporada no console do API Gateway para configurar a integração do método.

Como configurar uma integração de método

1. Selecione o recurso `/calc/{operand1}/{operand2}/{operator}` e, depois, escolha Criar método.
2. Em Tipo de método, selecione GET.
3. Em Tipo de integração, selecione Lambda.
4. Mantenha a opção Integração do proxy do Lambda desativada.
5. Em Função do Lambda, selecione a Região da AWS onde você criou a função do Lambda e insira **Calc**.
6. Mantenha o Tempo limite padrão ativado.
7. Escolha Criar método.

Agora, você vai criar um modelo de mapeamento para associar os três parâmetros de caminho de URL, declarados quando o recurso `/calc/{operand1}/{operand2}/{operator}` foi criado, a valores de propriedade designados no objeto JSON. Como os caminhos de URL devem ser codificados em URL, o operador de divisão deve ser especificado como `%2F` em vez de `/`. Esse modelo converte o `%2F` em `'/'` antes de transferi-lo para a função do Lambda.

Como criar um modelo de mapeamento

1. Na guia Solicitação de integração, em Configurações de solicitação de integração, selecione Editar.

2. Em Passagem do corpo da solicitação, selecione Quando não há modelos definidos (recomendado).
3. Selecione Modelos de mapeamento.
4. Em Tipo de conteúdo, insira **application/json**.
5. Em Corpo do modelo, insira o seguinte código:

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

6. Escolha Salvar.

Agora, é possível testar o método GET para verificar se ele foi configurado corretamente para invocar a função do Lambda e passar a saída original pela resposta de integração sem mapeamento.

Como testar o método **GET**

1. Selecione a guia Testar. Talvez seja necessário selecionar o botão de seta para a direita para mostrar a guia.
2. Para o Caminho, faça o seguinte:
 - a. Em operand1, insira **1**.
 - b. Em operand2, insira **1**.
 - c. Em operador, insira **+**.
3. Escolha Test (Testar).
4. O resultado deve ser semelhante ao seguinte:

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

Depois, você modelará a estrutura de dados da carga útil de resposta de método após o esquema `result`.

Por padrão, o corpo de resposta de método recebe um modelo vazio. Isso fará com que o corpo de resposta de integração seja repassado sem mapeamento. No entanto, quando você gerar um SDK para uma das linguagens de tipo forte, como Java ou Objective-C, os usuários do seu SDK receberão um objeto vazio como resultado. Para garantir que tanto o cliente REST quanto os clientes do SDK recebam o resultado desejado, você deve modelar os dados de resposta usando uma esquema predefinido. Aqui, você definirá um modelo para o corpo de resposta de método e para construir um modelo de mapeamento a fim de traduzir o corpo da resposta de integração no corpo da resposta de método.

Como criar uma resposta de método

1. Na guia Resposta de método, em Resposta 200, selecione Editar.
2. Em Corpo da resposta, selecione Adicionar modelo.
3. Em Tipo de conteúdo, insira **application/json**.
4. Em Modelo, selecione o resultado.
5. Escolha Salvar.

Definir o modelo `result` para o corpo da resposta de método garante que os dados da resposta serão convertidos no objeto de um determinado SDK. Para garantir que os dados de resposta de integração sejam mapeados de acordo, você precisará de um modelo de mapeamento.

Como criar um modelo de mapeamento

1. Na guia Resposta de integração, em Padrão - Resposta, selecione Editar.
2. Selecione Modelos de mapeamento.
3. Em Tipo de conteúdo, insira **application/json**.
4. Em Corpo do modelo, insira o seguinte código:

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

```
"output" : {  
  "c" : $inputRoot.c  
}  
}
```

5. Escolha Salvar.

Como testar o modelo de mapeamento

1. Selecione a guia Testar. Talvez seja necessário selecionar o botão de seta para a direita para mostrar a guia.
2. Para o Caminho, faça o seguinte:
 - a. Em operand1, insira **1**.
 - b. Em operand2, insira **2**.
 - c. Em operador, insira **+**.
3. Escolha Testar.
4. O resultado terá esta aparência:

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

Nesse momento, somente é possível chamar a API usando o atributo Testar no console do API Gateway. Para disponibilizá-la para os clientes, será necessário implantar a API. Certifique-se de reimplantar sua API sempre que você adicionar, modificar ou excluir um recurso ou método, atualizar um mapeamento de dados ou atualizar as configurações de estágio. Caso contrário, novos atributos ou atualizações não estarão disponíveis para clientes da API.

Para implantar a API

1. Escolha Implantar API.
2. Em Estágio, selecione Novo estágio.
3. Em Stage name (Nome do estágio), insira **Prod**.
4. (Opcional) Em Description (Descrição), insira uma descrição.
5. Escolha Implantar.
6. (Opcional) Em Detalhes do estágio, para Invocar URL, é possível selecionar o ícone de cópia para copiar o URL de invocação da API. Você pode usá-lo com ferramentas, como [Postman](#) e [cURL](#) para testar sua API.

Note

Reimplante a API sempre que você adicionar, modificar ou excluir um recurso ou um método, atualizar um mapeamento de dados ou atualizar as configurações de estágio. Caso contrário, novos recursos ou atualizações não estarão disponíveis para clientes de sua API.

Definições do OpenAPI da API demonstrativa integrada com uma função do 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\n\" : $inputRoot.a,\n  \"b\" : $inputRoot.b,\n  \"op\" : $inputRoot.op,\n  \"c\" : $inputRoot.c\n}"
        }
      }
    },
    "uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/arn:aws:lambda:us-west-2:123456789012:function:Calc/invocations",
    "passthroughBehavior": "when_no_templates",
    "httpMethod": "POST",
    "type": "aws"
  }
},
"/calc/{operand1}/{operand2}/{operator}": {
  "get": {
    "consumes": [
      "application/json"
    ],
    "produces": [
      "application/json"
    ],
    "parameters": [
      {
        "name": "operand2",

```

```

        "in": "path",
        "required": true,
        "type": "string"
    },
    {
        "name": "operator",
        "in": "path",
        "required": true,
        "type": "string"
    },
    {
        "name": "operand1",
        "in": "path",
        "required": true,
        "type": "string"
    }
],
"responses": {
    "200": {
        "description": "200 response",
        "schema": {
            "$ref": "#/definitions/Result"
        }
    }
},
"x-amazon-apigateway-integration": {
    "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
    "responses": {
        "default": {
            "statusCode": "200",
            "responseTemplates": {
                "application/json": "#set($inputRoot = $input.path('$'))\n{\n
\\\"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: Crie uma API REST como um proxy do Amazon S3

Como exemplo para mostrar o uso de uma API REST no API Gateway para o proxy do Amazon S3, esta seção descreve como criar e configurar uma API REST para expor as seguintes operações do Amazon S3:

- Expor GET no recurso raiz da API para [listar todos os buckets do Amazon S3 de um autor da chamada](#).
- Expor GET em um recurso Folder para [visualizar uma lista de todos os objetos no bucket do Amazon S3](#).
- Expor GET em um recurso Folder/Item para [visualizar ou baixar um objeto de um bucket do Amazon S3](#).

É recomendável importar a API de exemplo como um proxy do Amazon S3, conforme mostrado em [Definições do OpenAPI da API de amostra como um proxy do Amazon S3](#). Este exemplo contém

mais métodos expostos. Para obter instruções sobre como importar uma API usando a definição do OpenAPI, consulte [Desenvolver APIs REST usando OpenAPI no API Gateway](#).

Note

Para integrar sua API do API Gateway ao Amazon S3, é necessário escolher uma região onde os serviços API Gateway e Amazon S3 estejam disponíveis. Para obter a disponibilidade da região, consulte [Endpoints e cotas do Amazon API Gateway](#).

Tópicos

- [Configurar permissões do IAM para a API invocar ações do Amazon S3](#)
- [Criar recursos de API para representar recursos do Amazon S3](#)
- [Expor um método de API para listar os buckets do Amazon S3 do autor da chamada](#)
- [Expor métodos de API para acessar um bucket do Amazon S3](#)
- [Expor métodos de API para acessar um objeto do Amazon S3 em um bucket](#)
- [Definições do OpenAPI da API de amostra como um proxy do Amazon S3](#)
- [Chamar a API usando um cliente de API REST](#)

Configurar permissões do IAM para a API invocar ações do Amazon S3

Para permitir que a API invoque ações do Amazon S3, é necessário que as políticas do IAM apropriadas sejam associadas a um perfil do IAM. Neste exercício, você cria um perfil do IAM.

Como criar o perfil de execução do proxy de serviço da AWS

1. Faça login no Console de gerenciamento da AWS e abra o console do IAM em <https://console.aws.amazon.com/iam/>.
2. Escolha Perfis.
3. Selecione Criar perfil.
4. Selecione Serviço da AWS em Selecionar tipo de entidade confiável, selecione API Gateway e Permite que o API Gateway envie logs ao CloudWatch Logs.
5. Selecione Próximo e, depois, Próximo.
6. Em Role name (Nome da função), digite **APIGatewayS3ProxyPolicy** e escolha Create role (Criar função).

7. Na lista Roles (Funções), escolha a função que você acaba de criar. Talvez seja necessário rolar a página ou usar a barra de pesquisa para encontrar o perfil.
8. Para a função escolhida, selecione a guia Adicionar permissões.
9. Selecione Anexar políticas na lista suspensa.
10. Na barra de pesquisa, insira **AmazonS3FullAccess** e escolha Adicionar permissões.

 Note

Este tutorial usa uma política gerenciada em prol da simplicidade. Como prática recomendada, você deve criar sua própria política do IAM para conceder as permissões mínimas necessárias.

11. Anote o ARN do perfil recém-criado, você o usará posteriormente.

Criar recursos de API para representar recursos do Amazon S3

Use o recurso-raiz da API (/) como o contêiner de buckets do Amazon S3 de um chamador autenticado. Também crie recursos Folder e Item para representar um bucket do Amazon S3 específico e um determinado objeto do Amazon S3, respectivamente. O nome da pasta e a chave do objeto serão especificados, no formato de parâmetros de caminho como parte de uma URL de solicitação, pelo autor da chamada.

 Note

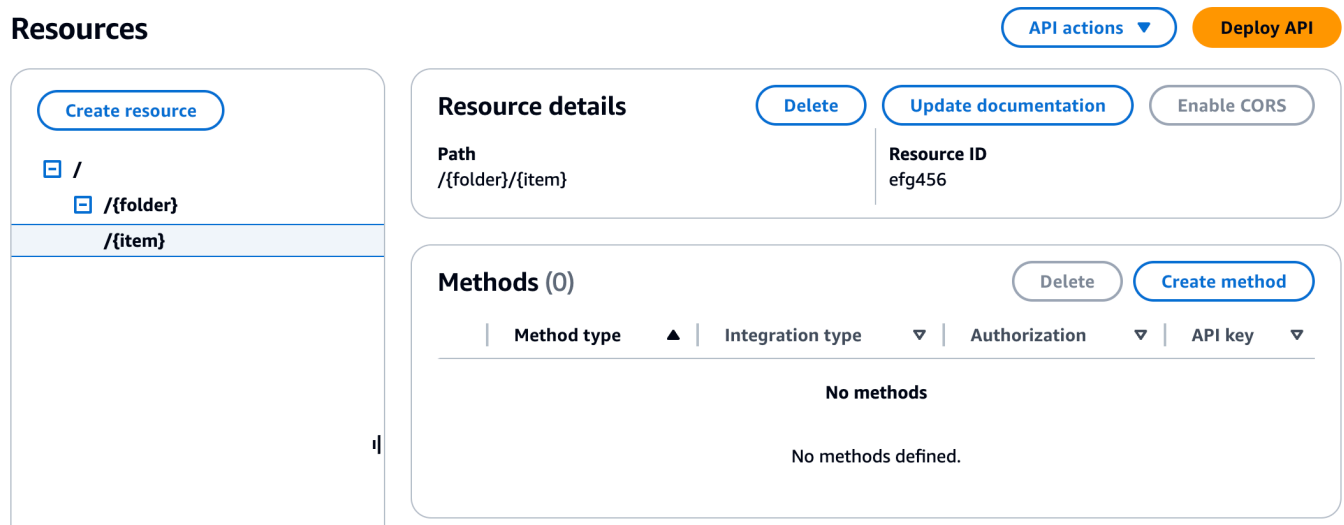
Ao acessar objetos cuja chave de objeto inclua / ou qualquer outro caractere especial, o caractere deverá ser codificado por URL. Por exemplo, `test/test.txt` deve ser codificado para `test%2Ftest.txt`.

Como criar um recurso de API que expõe os recursos de serviços do Amazon S3

1. Na mesma em Região da AWS onde você criou o bucket do Amazon S3, crie uma API chamada MyS3. O recurso raiz dessa API (/) representa o serviço Amazon S3. Nesta etapa, você vai criar dois recursos adicionais/{folder} e/{item}.
2. Selecione Criar recurso.
3. Mantenha Recurso proxy desativado.

4. Em Caminho do recurso, selecione `/`.
5. Em Resource Name (Nome do recurso), insira `{folder}`.
6. Mantenha CORS (Compartilhamento de recursos de origem cruzada) desmarcado.
7. Selecione Criar recurso.
8. Selecione o recurso `/ {folder}` e, depois, Criar recurso.
9. Use as etapas anteriores para criar um recurso secundário de `/ {folder}` chamado `{item}`.

A API final deverá ser semelhante à seguinte:



Expor um método de API para listar os buckets do Amazon S3 do autor da chamada

Obter a lista de buckets do Amazon S3 do autor da chamada envolve invocar a ação [GET Service](#) no Amazon S3. No recurso raiz do API, (`/`), crie o método GET. Configure o método GET para integrar-se ao Amazon S3 da maneira a seguir.

Para criar e inicializar o método **GET** `/` da API

1. Selecione o recurso `/` e Criar método.
2. Em tipo de método, selecione GET.
3. Em Tipo de integração, selecione AWS service (Serviço da AWS).
4. Para Região da AWS, selecione a Região da AWS onde você criou o bucket do Amazon S3.
5. Em AWS service (Serviço da AWS), selecione Amazon Simple Storage Service.
6. Mantenha o subdomínio da AWS em branco.

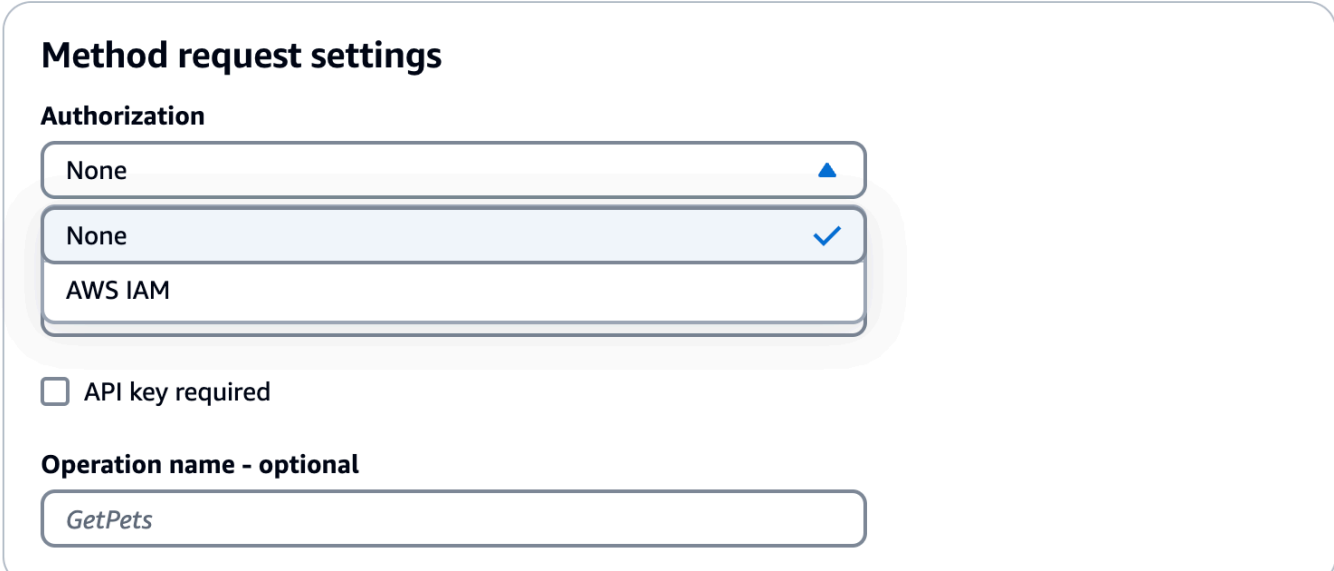
7. Em Método HTTP, selecione GET.
8. Em Tipo de ação, escolha Usar substituição de caminho.

Com a substituição de caminho, o API Gateway encaminha a solicitação do cliente para o Amazon S3 como a [solicitação no estilo de caminho da API REST do Amazon S3](#) correspondente em que um recurso do Amazon S3 é expresso pelo caminho de recurso do padrão s3-host-name/bucket/key. O API Gateway define o s3-host-name e transmite o bucket especificado pelo cliente e key do cliente para o Amazon S3.

9. Em Substituição de caminho, digite /.
10. Em Perfil de execução, digite o ARN do perfil para **APIGatewayS3ProxyPolicy**.
11. Escolha Configurações de solicitação de método.

Use as configurações de solicitação de método para controlar quem pode chamar esse método da sua API.

12. Em Autorização, no menu suspenso, selecione AWS_IAM.



Method request settings

Authorization

None

None

AWS IAM

☐ API key required

Operation name - optional

GetPets

13. Escolha Criar método.

Essa configuração integra a solicitação GET `https://your-api-host/stage/` do front-end com o GET `https://your-s3-host/` do backend.

Para que a API retorne respostas bem-sucedidas e exceções corretamente ao chamador, declare as respostas 200, 400 e 500 em Resposta de método. Use o mapeamento padrão para respostas 200

para que as respostas do back-end do código de status não declaradas aqui sejam retornadas ao chamador como respostas 200.

Como declarar tipos de resposta para o método **GET** /

1. Na guia Resposta de método, em Resposta 200, selecione Editar.
2. Selecione Adicionar usuário e faça o seguinte:
 - a. Em Nome do cabeçalho, insira **Content-Type**.
 - b. Escolha Add header (Adicionar cabeçalho).

Repita essas etapas para criar um cabeçalho **Timestamp** e um **Content-Length**.

3. Escolha Salvar.
4. Na guia Resposta de método, em Respostas do método, selecione Criar resposta.
5. Para o código de status HTTP, insira 400.

Você não vai definir nenhum cabeçalho para essa resposta.

6. Escolha Salvar.
7. Repita as etapas a seguir para criar a resposta 500.

Você não vai definir nenhum cabeçalho para essa resposta.

Como a resposta de integração bem-sucedida do Amazon S3 retorna a lista de buckets como uma carga útil XML, e a resposta de método padrão do API Gateway retorna uma carga útil JSON, você deve mapear o valor do parâmetro do cabeçalho Content-Type do back-end para sua contraparte do front-end. Caso contrário, o cliente receberá `application/json` para o tipo de conteúdo quando o corpo da resposta é na verdade uma string XML. O procedimento a seguir mostra como configurar isso. Além disso, mostre também para o cliente outros parâmetros de cabeçalho, como Date e Content-Length.

Para configurar mapeamentos de cabeçalho de resposta para o método GET /

1. Na guia Resposta de integração, em Padrão - Resposta, selecione Editar.
2. Para o cabeçalho Content-Length, insira **integration.response.header.Content-Length** para o valor do mapeamento.

3. Para o cabeçalho Content-Type, insira **integration.response.header.Content-Type** para o valor do mapeamento.
4. Para o cabeçalho Carimbo de data e hora, insira **integration.response.header.Date** para o valor do mapeamento.
5. Escolha Salvar. O resultado deve ser semelhante ao seguinte:

< **Method request** **Integration request** **Integration response** **Met** >

Integration responses

[Create response](#)

Default - Response

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

HTTP status regex [Info](#)
-

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

Method response status code
200

Default mapping
True

Header mappings (3)

< **1** >

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

Mapping templates (0)

No templates

You don't have any mapping templates.

6. Na guia Resposta de integração, em Respostas de integração, selecione Criar resposta.

7. Em HTTP status regex (Regex de status HTTP), insira **4\d{2}**. Isso associa todos os códigos de status de resposta HTTP 4xx à resposta do método.
8. Em Código de status de resposta do método, selecione **400**.
9. Escolha Criar.
10. Repita as etapas a seguir para criar uma resposta de integração para a resposta do método 500. Em HTTP status regex (Regex de status HTTP), insira **5\d{2}**.

Como prática recomendada, teste a API que você configurou até agora.

Como testar o método **GET** /

1. Selecione a guia Testar. Talvez seja necessário selecionar o botão de seta para a direita para mostrar a guia.
2. Escolha Testar. O resultado deve ser algo semelhante à seguinte imagem:

Method request

Integration request

Integration response

Method response

Test

Test method

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

Query strings

`param1=value1&param2=value2`

Headers

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

`header1:value1`
`header2:value2`

Client certificate

No client certificates have been generated.

Test

/ - GET method test results

Request

/

Latency ms

74

Status

200

Response body

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

Expor métodos de API para acessar um bucket do Amazon S3

Para trabalhar com um bucket do Amazon S3, exponha o método GET no recurso `/{{folder}}` para listar objetos em um bucket. As instruções são semelhantes às descritas em [Expor um método de API para listar os buckets do Amazon S3 do autor da chamada](#). Para conhecer mais métodos, é possível importar a API de exemplo aqui, [Definições do OpenAPI da API de amostra como um proxy do Amazon S3](#).

Como expor o método GET em um recurso de pasta

1. Selecione o recurso `/{{folder}}` e, depois, Criar método.
2. Em tipo de método, selecione GET.

3. Em Tipo de integração, selecione AWS service (Serviço da AWS).
4. Para Região da AWS, selecione a Região da AWS onde você criou o bucket do Amazon S3.
5. Em AWS service (Serviço da AWS), selecione Amazon Simple Storage Service.
6. Mantenha o subdomínio da AWS em branco.
7. Em Método HTTP, selecione GET.
8. Em Tipo de ação, escolha Usar substituição de caminho.
9. Em Substituição de caminho, digite **{bucket}**.
10. Em Perfil de execução, digite o ARN do perfil para **APIGatewayS3ProxyPolicy**.
11. Escolha Criar método.

Você vai definir os parâmetros de caminho `{folder}` no URL do endpoint do Amazon S3. É necessário associar o parâmetro de caminho `{folder}` da solicitação de método ao parâmetro de caminho `{bucket}` da solicitação de integração.

Como associar **{folder}** a **{bucket}**

1. Na guia Solicitação de integração, em Configurações de solicitação de integração, selecione Editar.
2. Selecione Parâmetros de caminho de URL e, depois, Adicionar parâmetro de caminho.
3. Em Nome, digite **bucket**.
4. Em Mapeado de, insira **method.request.path.folder**.
5. Escolha Salvar.

Agora, você vai testar a API.

Como testar o método **/ {folder} GET**.

1. Selecione a guia Testar. Talvez seja necessário selecionar o botão de seta para a direita para mostrar a guia.
2. Em Caminho, em pasta, insira o nome do bucket.
3. Escolha Testar.

O resultado do teste conterá uma lista de objetos no bucket.

Test method

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

Path

folder

Query strings

Headers

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

Client certificate

No client certificates have been generated.

Test

Info

/{folder} - GET method test results

Request

/amzn-s3-demo-bucket

Latency ms

59

Status

200

Response body

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

Expor métodos de API para acessar um objeto do Amazon S3 em um bucket

O Amazon S3 oferece suporte a ações GET, DELETE, HEAD, OPTIONS, POST e PUT para acessar e gerenciar objetos em um determinado bucket. Neste tutorial, você vai expor um método GET no recurso {folder}/{item} para obter uma imagem de um bucket. Para conhecer mais aplicações do recurso {folder}/{item}, consulte a API de exemplo, [Definições do OpenAPI da API de amostra como um proxy do Amazon S3](#).

Como expor o método GET em um recurso de item

1. Selecione o recurso /{item} e, depois, Criar método.
2. Em tipo de método, selecione GET.
3. Em Tipo de integração, selecione AWS service (Serviço da AWS).
4. Para Região da AWS, selecione a Região da AWS onde você criou o bucket do Amazon S3.

5. Em AWS service (Serviço da AWS), selecione Amazon Simple Storage Service.
6. Mantenha o subdomínio da AWS em branco.
7. Em Método HTTP, selecione GET.
8. Em Tipo de ação, escolha Usar substituição de caminho.
9. Em Substituição de caminho, insira `{bucket}/{object}`.
10. Em Perfil de execução, digite o ARN do perfil para **APIGatewayS3ProxyPolicy**.
11. Escolha Criar método.

Você vai definir os parâmetros de caminho `{folder}` e o `{item}` no URL do endpoint do Amazon S3. É necessário associar o parâmetro de caminho da solicitação de método ao parâmetro de caminho da solicitação de integração.

Nesta etapa, faça o seguinte:

- Mapeie o parâmetro de caminho `{folder}` da solicitação de método ao parâmetro de caminho `{bucket}` da solicitação de integração.
- Mapeie o parâmetro de caminho `{item}` da solicitação de método ao parâmetro de caminho `{object}` da solicitação de integração.

Como associar **{folder}** a **{bucket}** e **{item}** a **{object}**

1. Na guia Solicitação de integração, em Configurações de solicitação de integração, selecione Editar.
2. Selecione Parâmetros de caminho de URL.
3. Selecione Adicionar parâmetro de caminho.
4. Em Nome, digite **bucket**.
5. Em Mapeado de, insira **method.request.path.folder**.
6. Selecione Adicionar parâmetro de caminho.
7. Em Nome, digite **object**.
8. Em Mapeado de, insira **method.request.path.item**.
9. Escolha Salvar.

Como testar o método `/folder/object GET`.

1. Selecione a guia Testar. Talvez seja necessário selecionar o botão de seta para a direita para mostrar a guia.
2. Em Caminho, em pasta, insira o nome do bucket.
3. Em Caminho, em item, insira o nome de um item.
4. Escolha Testar.

O corpo da resposta incluirá o conteúdo do item.

[illegible]

A solicitação exibe corretamente o texto sem formatação (“Hello world”) como o conteúdo do arquivo especificado (test.txt) no bucket do Amazon S3 especificado (amzn-s3-demo-bucket).

Para fazer download ou upload de arquivos binários, que no API Gateway é qualquer conteúdo que não seja JSON codificado em utf-8, são necessárias configurações de API adicionais. Isto é descrito da seguinte forma:

Para fazer download ou upload de arquivos binários do S3

1. Registre os tipos de mídia do arquivo afetado para `binaryMediaTypes` da API. Você pode fazer isso no console:
 - a. Selecione Configurações para a API.
 - b. Em Tipos de mídia binária, selecione Adicionar tipo de mídia binária.
 - c. Selecione Adicionar tipo de mídia binária e insira o tipo de mídia necessário, por exemplo, `image/png`.
 - d. Selecione Save changes (Salvar alterações) para salvar a configuração.
2. Adicione o cabeçalho `Content-Type` (para upload) e/ou `Accept` (para download) à solicitação de método para exigir que o cliente especifique o tipo de mídia binário necessário e o mapeie para a solicitação de integração.
3. Defina Tratamento de conteúdo como `Passthrough` na solicitação de integração (para upload) e em uma resposta de integração (para download). Certifique-se de que nenhum modelo de mapeamento está definido para o tipo de conteúdo afetado. Para obter mais informações, consulte [the section called “Transformações de dados”](#).

O tamanho máximo da carga é 10 MB. Consulte [Cotas para configurar e executar uma API REST no API Gateway](#).

Certifique-se de que os arquivos no Amazon S3 tenham os tipos de conteúdo corretos adicionados como metadados de arquivos. Para a transmissão de conteúdo de mídia, talvez também seja necessário adicionar `Content-Disposition:inline` aos metadados.

Para obter mais informações sobre o suporte binário no API Gateway, consulte [Conversões de tipo de conteúdo no API Gateway](#).

Definições do OpenAPI da API de amostra como um proxy do Amazon S3

As definições da OpenAPI a seguir descrevem uma API que funciona como um proxy do Amazon S3. Essa API contém mais operações do Amazon S3 do que a API que você criou no tutorial. Os seguintes métodos estão expostos nas definições da OpenAPI:

- Expor GET no recurso raiz da API para [listar todos os buckets do Amazon S3 de um autor da chamada](#).
- Expor GET em um recurso Folder para [visualizar uma lista de todos os objetos no bucket do Amazon S3](#).
- Expor PUT em um recurso Folder para [adicionar um bucket ao Amazon S3](#).
- Expor DELETE em um recurso Folder para [remover um bucket do Amazon S3](#).
- Expor GET em um recurso Folder/Item para [visualizar ou baixar um objeto de um bucket do Amazon S3](#).
- Expor PUT em um recurso Folder/Item para [fazer upload de um objeto em um bucket do Amazon S3](#).
- Expor HEAD em um recurso Folder/Item para [obter um objeto de metadados em um bucket do Amazon S3](#).
- Expor DELETE em um recurso Folder/Item para [remover um objeto de um bucket do Amazon S3](#).

Para obter instruções sobre como importar uma API usando a definição do OpenAPI, consulte [Desenvolver APIs REST usando OpenAPI no API Gateway](#).

Para obter instruções sobre como criar uma API simples, consulte [Tutorial: Crie uma API REST como um proxy do Amazon S3](#).

Para saber como invocar essa API usando o [Postman](#), que é compatível com a autorização do IAM AWS, consulte [Chamar a API usando um cliente de API 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"
    }
  }
}
}
}

```

Chamar a API usando um cliente de API REST

Para fornecer um tutorial abrangente, agora mostramos como chamar a API usando o [Postman](#), que oferece suporte à autorização do IAM da AWS.

Como chamar nossa API de proxy do Amazon S3 usando Postman

1. Implante ou reimplante a API. Anote a URL base da API que é exibida ao lado de Invocar URL na parte superior de Editor de estágio.
2. Inicie o Postman.
3. Escolha Autorização e AWS Signature. Insira o ID de chave de acesso e a chave de acesso secreta do usuário do IAM nos campos de entrada AccessKey e SecretKey, respectivamente. Insira a Região da AWS onde sua API está implantada na caixa de texto Região da AWS. Insira `execute-api` no campo de entrada Nome do serviço.

É possível criar um par de chaves da guia Security Credentials (Credenciais de segurança) da conta de usuário do IAM no Console de Gerenciamento do IAM.

4. Para adicionar um bucket denominado `amzn-s3-demo-bucket` à conta do Amazon S3 na região `{region}`:
 - a. Escolha PUT na lista suspensa de métodos e digite a URL do método (`https://api-id.execute-api.aws-region.amazonaws.com/stage/folder-name`)
 - b. Defina o valor do cabeçalho Content-Type como `application/xml`. Talvez seja necessário excluir todos os cabeçalhos existentes antes de definir o tipo de conteúdo.
 - c. Escolha o item de menu Corpo e digite o seguinte fragmento de XML como o corpo da solicitação:

```
<CreateBucketConfiguration>
  <LocationConstraint>{region}</LocationConstraint>
</CreateBucketConfiguration>
```
 - d. Escolha Enviar para enviar a solicitação. Se for bem-sucedido, você receberá uma resposta `200 OK` com uma carga vazia.
5. Para adicionar um arquivo de texto a um bucket, siga as instruções acima. Se você especificar o nome de um bucket de `amzn-s3-demo-bucket` como `{folder}` e o nome de um arquivo de `Readme.txt` como `{item}` no URL e fornecer uma string de texto de **Hello, World!** como conteúdos de arquivos (tornando-a, dessa forma, a carga da solicitação), a solicitação se tornará

```
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!
```

Se tudo der certo, você receberá uma resposta 200 OK com uma carga vazia.

6. Para obter o conteúdo do arquivo `Readme.txt` que acabamos de adicionar ao bucket `amzn-s3-demo-bucket`, faça uma solicitação GET como a seguinte:

```
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
```

Se for bem-sucedido, você receberá uma resposta 200 OK com uma string de texto `Hello, World!` como carga.

7. Para listar itens no bucket `amzn-s3-demo-bucket`, envie a seguinte solicitação:

```
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
```

Se for bem-sucedido, você receberá uma resposta 200 OK com uma carga XML mostrando um único item no bucket especificado, a menos que tenha adicionado mais arquivos ao bucket antes de enviar essa solicitação.

```
<?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 carregar ou baixar uma imagem, você precisa configurar a manipulação de conteúdo para CONVERT_TO_BINARY.

Tutorial: Criar uma API REST como um proxy do Amazon Kinesis

Esta página descreve como criar e configurar uma API REST com uma integração do tipo AWS para acessar o Kinesis.

Note

Para integrar sua API do API Gateway ao Kinesis, é necessário escolher uma região onde os serviços API Gateway e Kinesis estejam disponíveis. Para saber a disponibilidade da região, consulte [Endpoints e cotas de serviço](#).

Para fins de ilustração, criamos uma API de exemplo para permitir que um cliente faça o seguinte:

1. Listar os streams disponíveis do usuário no Kinesis
2. Criar, descrever ou excluir um fluxo especificado
3. Leia registros de dados ou escreva registros de dados no fluxo especificado

Para realizar as tarefas anteriores, a API expõe métodos em vários recursos para invocar o seguinte, respectivamente:

1. A ação `ListStreams` no Kinesis
2. A ação `CreateStream`, `DescribeStream` ou `DeleteStream`
3. A ação `GetRecords` ou `PutRecords` (incluindo `PutRecord`) no Kinesis

Especificamente, construiremos a API da seguinte maneira:

- Expondo um método HTTP GET no recurso `/streams` da API e integrando o método à ação [ListStreams](#) no Kinesis para listar os streams na conta do autor da chamada.
- Expondo um método HTTP POST no recurso `/streams/{stream-name}` da API e integrando esse método à ação [CreateStream](#) no Kinesis para criar um stream nomeado na conta do autor da chamada.
- Expondo um método HTTP GET no recurso `/streams/{stream-name}` da API e integrando esse método à ação [DescribeStream](#) no Kinesis para descrever um stream nomeado na conta do autor da chamada.
- Expondo um método HTTP DELETE no recurso `/streams/{stream-name}` da API e integrando o método à ação [DeleteStream](#) no Kinesis para excluir um stream na conta do autor da chamada.
- Expondo um método HTTP PUT no recurso `/streams/{stream-name}/record` da API e integrando o método à ação [PutRecord](#) no Kinesis. Isso permite que o cliente adicione um único registro de dados ao fluxo nomeado.
- Expondo um método HTTP PUT no recurso `/streams/{stream-name}/records` da API e integrando o método à ação [PutRecords](#) no Kinesis. Isso permite que o cliente adicione uma lista de registros de dados ao fluxo nomeado.
- Expondo um método HTTP GET no recurso `/streams/{stream-name}/records` da API e integrando esse método à ação [GetRecords](#) no Kinesis. Isso permite que o cliente liste registros de dados no fluxo nomeado com um iterador de fragmentos especificado. Um iterador de fragmentos especifica a posição do fragmento a partir da qual começar a ler os registros de dados sequencialmente.

- Expondo um método HTTP GET no recurso `/streams/{stream-name}/sharditerator` da API e integrando esse método à ação [GetShardIterator](#) no Kinesis. Esse método auxiliar deve ser fornecido à ação `ListStreams` no Kinesis.

É possível aplicar as instruções apresentadas aqui a outras ações do Kinesis. Para obter a lista completa das ações do Kinesis, consulte [Referência de API do Amazon Kinesis](#).

Em vez de usar o console do API Gateway para criar a API demonstrativa, é possível importar a API demonstrativa para o API Gateway usando a [API de importação](#) do API Gateway. Para obter informações sobre como usar o recurso Import API, consulte [Desenvolver APIs REST usando OpenAPI no API Gateway](#).

Criar uma função e política do IAM para a API acessar o Kinesis

Para que a API invoque ações do Kinesis, é necessário ter as políticas do IAM apropriadas anexadas a um perfil do IAM. Neste exercício, você cria um perfil do IAM.

Como criar o perfil de execução do proxy de serviço da AWS

1. Faça login no Console de gerenciamento da AWS e abra o console do IAM em <https://console.aws.amazon.com/iam/>.
2. Escolha Perfis.
3. Selecione Criar perfil.
4. Selecione Serviço da AWS em Selecionar tipo de entidade confiável, selecione API Gateway e Permita que o API Gateway envie logs ao CloudWatch Logs.
5. Selecione Próximo e, depois, Próximo.
6. Em Role name (Nome da função), digite **APIGatewayKinesisProxyPolicy** e escolha Create role (Criar função).
7. Na lista Roles (Funções), escolha a função que você acaba de criar. Talvez seja necessário rolar a página ou usar a barra de pesquisa para encontrar o perfil.
8. Para a função escolhida, selecione a guia Adicionar permissões.
9. Selecione Anexar políticas na lista suspensa.
10. Na barra de pesquisa, insira **AmazonKinesisFullAccess** e selecione Adicionar permissões.

 Note

Este tutorial usa uma política gerenciada em prol da simplicidade. Como prática recomendada, você deve criar sua própria política do IAM para conceder as permissões mínimas necessárias.

11. Anote o ARN do perfil recém-criado, você o usará posteriormente.

Criar uma API como um proxy do Kinesis

Use as etapas a seguir para criar a API no console do API Gateway.

Como criar uma API como um proxy de serviço da AWS para o Kinesis

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Se esta for a primeira vez que você usa o API Gateway, você verá uma página com os recursos do serviço. Em REST API, escolha Build (Criar). Quando o pop-up Create Example API (Criar API de exemplo) for exibido, escolha OK.

Se essa não for a primeira vez que você usa o API Gateway, escolha Create API (Criar API). Em REST API, escolha Build (Criar).

3. Selecione New API (Nova API).
4. Em API name (Nome da API), insira **KinesisProxy**. Mantenha os valores padrão para todos os outros campos.
5. (Opcional) Em Description (Descrição), insira uma descrição.
6. Em Tipo de endereço IP, selecione IPv4.
7. Selecione Create API (Criar API).

Após a criação da API, o console do API Gateway exibe a página Resources (Recursos), que contém apenas o recurso raiz (/) da API.

Listar streams no Kinesis

O Kinesis é compatível com a ação `ListStreams` com a seguinte chamada da API 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>

{
  ...
}
```

Na solicitação de API REST acima, a ação é especificada no parâmetro `Action` da consulta. Como alternativa, você pode especificar a ação em um cabeçalho `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

{
  ...
}
```

Neste tutorial, usamos o parâmetro de consulta para especificar a ação.

Para expor uma ação do Kinesis na API, adicione um recurso `/streams` à raiz da API. Depois, defina um método GET no recurso e integre o método com a ação `ListStreams` do Kinesis.

O procedimento a seguir descreve como listar streams do Kinesis usando o console do API Gateway.

Como listar streams do Kinesis usando o console do API Gateway

1. Selecione o recurso `/` e, depois, escolha Criar recurso.
2. Em Resource Name (Nome do recurso), insira **streams**.
3. Mantenha CORS (Compartilhamento de recursos de origem cruzada) desativado.
4. Selecione Criar recurso.
5. Selecione o recurso `/streams` e, depois, Criar método e faça o seguinte:

- a. Em Tipo de método, selecione GET.

 Note

O verbo HTTP para um método invocado por um cliente pode ser diferente do verbo HTTP para uma integração exigida pelo backend. Aqui, selecionamos GET porque a listagem de fluxos é intuitivamente uma operação READ.

- b. Em Tipo de integração, selecione Serviço da AWS.
- c. Em Região da AWS, selecione a Região da AWS onde você criou o fluxo do Kinesis.
- d. Em AWS service (Serviço da AWS), selecione Kinesis.
- e. Mantenha o subdomínio da AWS em branco.
- f. Em Método HTTP, escolha POST.

 Note

Aqui, escolhemos POST porque o Kinesis exige que a ação `ListStreams` seja invocada com ele.

- g. Em Tipo de ação, selecione Usar nome da ação.
 - h. Em Nome da ação, insira **ListStreams**.
 - i. Em Perfil de execução, digite o ARN para o perfil de execução.
 - j. Deixe o padrão de Passagem para Manuseio de conteúdo.
 - k. Escolha Criar método.
6. Na guia Solicitação de integração, em Configurações de solicitação de integração, selecione Editar.
 7. Em Passagem do corpo da solicitação, selecione Quando não há modelos definidos (recomendado).
 8. Selecione Parâmetros de cabeçalhos de solicitações de URL e faça o seguinte:
 - a. Selecione Adicionar parâmetro de cabeçalhos de solicitação.
 - b. Em Nome, digite **Content-Type**.
 - c. Em Mapeado de, insira **'application/x-amz-json-1.1'**.

Usamos um mapeamento de parâmetros de solicitação para definir o cabeçalho Content-Type como o valor estático de 'application/x-amz-json-1.1' para informar ao Kinesis que a entrada é de uma versão específica do JSON.

9. Selecione Modelos de mapeamento, Adicionar modelo de mapeamento e faça o seguinte:
 - a. Em Tipo de conteúdo, insira **application/json**.
 - b. Em Corpo do modelo, insira **{}**.
 - c. Escolha Salvar.

A solicitação [ListStreams](#) requer uma carga com o seguinte formato JSON:

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

No entanto, as propriedades são opcionais. Para usar os valores padrão, optamos por uma carga JSON vazia aqui.

10. Teste o método GET no recurso /streams para invocar a ação ListStreams no Kinesis:

Selecione a guia Testar. Talvez seja necessário selecionar o botão de seta para a direita para mostrar a guia.

Selecione Testar para testar o método.

Se você já criou dois streams chamados "myStream" e "yourStream" no Kinesis, o teste bem-sucedido retornará uma resposta 200 OK contendo a seguinte carga útil:

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

```
}
```

Criar, descrever e excluir um stream no Kinesis

As tarefas de criar, descrever e excluir um stream no Kinesis envolvem fazer as seguintes solicitações de API REST do 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 construir a API para aceitar a entrada necessária como uma carga de JSON da solicitação de método e passar essa carga diretamente à solicitação de integração. No entanto, para fornecer mais exemplos do mapeamento de dados entre solicitações de método e integração e respostas de método e integração, criamos nossa API de uma maneira um pouco diferente.

Expomos os métodos HTTP GET, POST e Delete em um recurso de Stream a ser nomeado. Usamos a variável de caminho {stream-name} como o espaço reservado do recurso de stream e integramos esses métodos de API às ações DescribeStream, CreateStream e DeleteStream do Kinesis, respectivamente. Exigimos que o cliente passe outros dados de entrada como cabeçalhos, parâmetros de consulta ou a carga de uma solicitação de método. Fornecemos modelos de mapeamento para transformar os dados na carga da solicitação de integração necessária.

Como criar o recurso {stream-name}

1. Selecione o recurso /streams e Criar recurso.
2. Mantenha Recurso proxy desativado.
3. Em Caminho do recurso, selecione /streams.
4. Em Resource Name (Nome do recurso), insira **{stream-name}**.
5. Mantenha CORS (Compartilhamento de recursos de origem cruzada) desativado.
6. Selecione Criar recurso.

Para configurar e testar o método GET em um recurso de fluxo

1. Selecione o recurso /{stream-name} e Criar método.
2. Em Tipo de método, selecione GET.
3. Em Tipo de integração, selecione Serviço da AWS.
4. Em Região da AWS, selecione a Região da AWS onde você criou o fluxo do Kinesis.
5. Em AWS service (Serviço da AWS), selecione Kinesis.
6. Mantenha o subdomínio da AWS em branco.
7. Em Método HTTP, escolha POST.
8. Em Tipo de ação, selecione Usar nome da ação.

9. Em Nome da ação, insira **DescribeStream**.
10. Em Perfil de execução, digite o ARN para o perfil de execução.
11. Deixe o padrão de Passagem para Manuseio de conteúdo.
12. Escolha Criar método.
13. Na seção Solicitação de integração, adicione os seguintes parâmetros de cabeçalhos de solicitação de URL:

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

A tarefa segue o mesmo procedimento para configurar o mapeamento do parâmetro de solicitação para o método GET `/streams`.

14. Adicione o seguinte modelo de mapeamento de corpo para mapear dados na solicitação do método GET `/streams/{stream-name}` à solicitação de integração POST `/?Action=DescribeStream`:

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

Esse modelo de mapeamento gera a carga de solicitação de integração necessária para a ação `DescribeStream` do Kinesis do valor do parâmetro do caminho `stream-name` da solicitação do método.

15. Para testar o método GET `/stream/{stream-name}` para invocar a ação `DescribeStream` no Kinesis, selecione a guia Testar.
16. Em Caminho, em `stream-name`, insira o nome de um fluxo existente do Kinesis.
17. Escolha Testar. Se o teste for bem-sucedido, uma resposta 200 OK será retornada com uma carga semelhante à seguinte:

```
{
  "StreamDescription": {
    "HasMoreShards": false,
    "RetentionPeriodHours": 24,
    "Shards": [
      {
        "HashKeyRange": {
          "EndingHashKey": "68056473384187692692674921486353642290",
          "StartingHashKey": "0"
        }
      }
    ]
  }
}
```

```

    },
    "SequenceNumberRange": {
      "StartingSequenceNumber":
"49559266461454070523309915164834022007924120923395850242"
    },
    "ShardId": "shardId-000000000000"
  },
  ...
  {
    "HashKeyRange": {
      "EndingHashKey": "340282366920938463463374607431768211455",
      "StartingHashKey": "272225893536750770770699685945414569164"
    },
    "SequenceNumberRange": {
      "StartingSequenceNumber":
"49559266461543273504104037657400164881014714369419771970"
    },
    "ShardId": "shardId-000000000004"
  }
],
"StreamARN": "arn:aws:kinesis:us-east-1:12345678901:stream/myStream",
"StreamName": "myStream",
"StreamStatus": "ACTIVE"
}
}

```

Depois de implantar a API, você poderá fazer uma solicitação REST com base neste método de 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 e testar o método POST em um recurso de fluxo

1. Selecione o recurso `/stream-name` e Crie método.

2. Em Tipo de método, selecione POST.
3. Em Tipo de integração, selecione Serviço da AWS.
4. Em Região da AWS, selecione a Região da AWS onde você criou o fluxo do Kinesis.
5. Em AWS service (Serviço da AWS), selecione Kinesis.
6. Mantenha o subdomínio da AWS em branco.
7. Em Método HTTP, escolha POST.
8. Em Tipo de ação, selecione Usar nome da ação.
9. Em Nome da ação, insira **CreateStream**.
10. Em Perfil de execução, digite o ARN para o perfil de execução.
11. Deixe o padrão de Passagem para Manuseio de conteúdo.
12. Escolha Criar método.
13. Na seção Solicitação de integração, adicione os seguintes parâmetros de cabeçalhos de solicitação de URL:

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

A tarefa segue o mesmo procedimento para configurar o mapeamento do parâmetro de solicitação para o método GET `/streams`.

14. Adicione o seguinte modelo de mapeamento de corpo para mapear dados na solicitação do método POST `/streams/{stream-name}` à solicitação de integração POST `/?Action=CreateStream`:

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

No modelo de mapeamento anterior, definimos `ShardCount` como um valor fixo de 5 se o cliente não especificar um valor na carga da solicitação do método.

15. Para testar o método POST `/stream/{stream-name}` para invocar a ação `CreateStream` no Kinesis, selecione a guia Testar.
16. Em Caminho, em `stream-name`, insira o nome de um novo fluxo do Kinesis.
17. Escolha Testar. Se o teste for bem-sucedido, uma resposta 200 OK será retornado sem dados.

Depois de implantar a API, também será possível fazer uma solicitação de API REST no método POST em um recurso Stream para invocar a ação `CreateStream` no 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
}
```

Configure e teste o método DELETE em um recurso de fluxo

1. Selecione o recurso `/stream-name` e Criar método.
2. Em Tipo de método, selecione DELETE.
3. Em Tipo de integração, selecione Serviço da AWS.
4. Em Região da AWS, selecione a Região da AWS onde você criou o fluxo do Kinesis.
5. Em AWS service (Serviço da AWS), selecione Kinesis.
6. Mantenha o subdomínio da AWS em branco.
7. Em Método HTTP, escolha POST.
8. Em Tipo de ação, selecione Usar nome da ação.
9. Em Nome da ação, insira **DeleteStream**.
10. Em Perfil de execução, digite o ARN para o perfil de execução.
11. Deixe o padrão de Passagem para Manuseio de conteúdo.
12. Escolha Criar método.
13. Na seção Solicitação de integração, adicione os seguintes parâmetros de cabeçalhos de solicitação de URL:

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

A tarefa segue o mesmo procedimento para configurar o mapeamento do parâmetro de solicitação para o método GET `/streams`.

14. Adicione o seguinte modelo de mapeamento de corpo para mapear dados na solicitação do método DELETE `/streams/{stream-name}` à solicitação de integração correspondente de POST `/?Action=DeleteStream`:

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

Esse modelo de mapeamento gera a entrada necessária para a ação DELETE `/streams/{stream-name}` no nome do caminho da URL de `stream-name` fornecido pelo cliente.

15. Para testar o método DELETE `/stream/{stream-name}` para invocar a ação `DeleteStream` no Kinesis, selecione a guia Testar.
16. Em Caminho, em `stream-name`, insira o nome de um fluxo existente do Kinesis.
17. Escolha Testar. Se o teste for bem-sucedido, uma resposta 200 OK será retornado sem dados.

Depois de implantar a API, você também poderá fazer a solicitação de API REST a seguir no método DELETE em um recurso Stream para chamar a ação `DeleteStream` no 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

{}
```

Obter registros de e adicionar registros a um stream no Kinesis

Depois de criar um stream no Kinesis, você poderá adicionar registros de dados ao stream e ler os dados desse stream. Adicionar registros de dados envolve chamar a ação [PutRecords](#) ou [PutRecord](#) no Kinesis. O primeiro adiciona vários registros, enquanto o último adiciona um único registro ao fluxo.

```
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"
}
```

ou

```
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"
}
```

Aqui, `StreamName` identifica o fluxo de destino para adicionar registros. `StreamName`, `Data` e `PartitionKey` são dados de entrada necessários. No nosso exemplo, podemos usar os valores

padrão para todos os dados de entrada opcionais e não especificaremos explicitamente valores para eles na entrada para a solicitação de método.

Ler dados no Kinesis equivale a chamar a ação [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
}
```

Aqui, o stream de origem do qual estamos obtendo registros é especificado no valor `ShardIterator` necessário, conforme indicado na ação a seguir do Kinesis para obter um iterador de fragmentos:

```
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

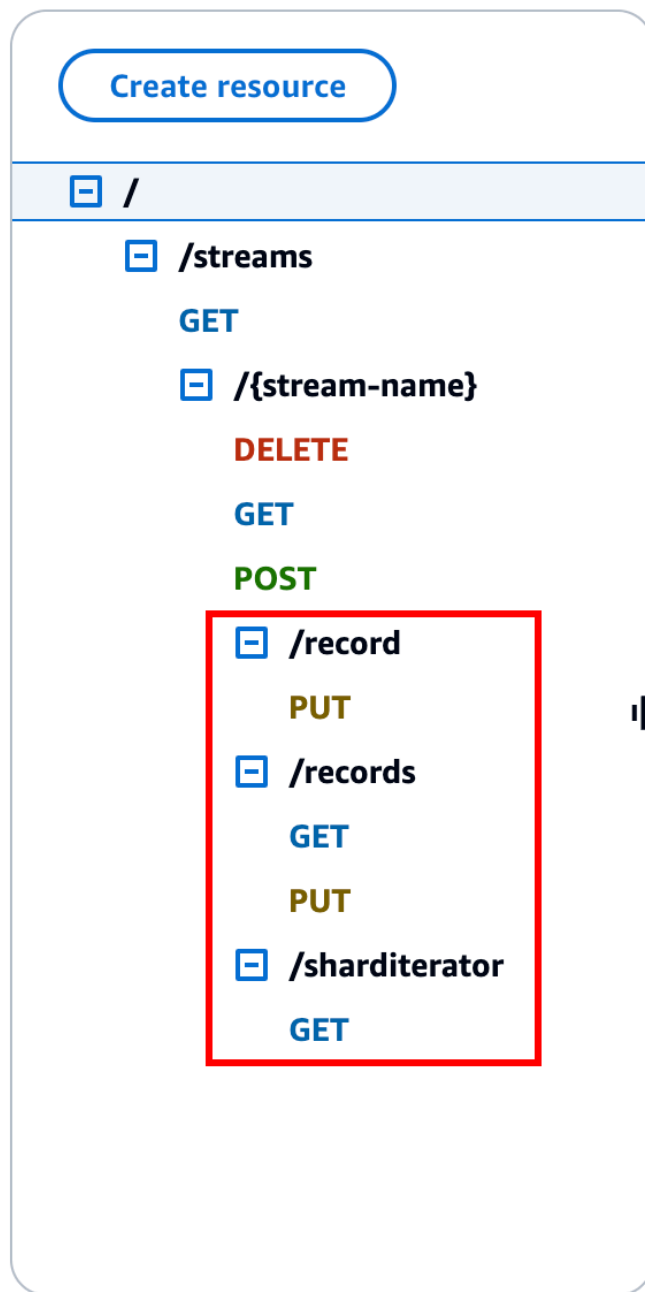
{
  "ShardId": "string",
  "ShardIteratorType": "string",
  "StartingSequenceNumber": "string",
  "StreamName": "string"
}
```

Para as ações `GetRecords` e `PutRecords`, expomos os métodos GET e PUT, respectivamente, em um recurso `/records` que está anexado a um recurso de fluxo nomeado (`/stream-name`). Da mesma forma, expomos a ação `PutRecord` como um método PUT em um recurso `/record`.

Como a ação `GetRecords` usa como entrada um valor `ShardIterator`, que é obtido ao chamar a ação auxiliar `GetShardIterator`, expomos um método auxiliar `GET` em um recurso `ShardIterator (/sharditerator)`.

Como criar os recursos `/record`, `/records` e `/sharditerator`

1. Selecione o recurso `/stream-name` e Criar recurso.
2. Mantenha Recurso proxy desativado.
3. Em Caminho do recurso, selecione `/stream-name`.
4. Em Resource Name (Nome do recurso), insira **record**.
5. Mantenha CORS (Compartilhamento de recursos de origem cruzada) desativado.
6. Selecione Criar recurso.
7. Repita as etapas anteriores para criar um recurso `/records` e um `/sharditerator`. A API final deve ter a seguinte aparência:



Os quatro procedimentos a seguir descrevem como configurar cada um dos métodos, como mapear dados de solicitações de método para solicitações de integração e como testar os métodos.

Como configurar e testar o método **PUT /streams/{stream-name}/record** para invocar **PutRecord** no Kinesis:

1. Selecione o recurso **/record** e Crie método.

2. Em Tipo de método, selecione PUT.
3. Em Tipo de integração, selecione Serviço da AWS.
4. Em Região da AWS, selecione a Região da AWS onde você criou o fluxo do Kinesis.
5. Em AWS service (Serviço da AWS), selecione Kinesis.
6. Mantenha o subdomínio da AWS em branco.
7. Em Método HTTP, escolha POST.
8. Em Tipo de ação, selecione Usar nome da ação.
9. Em Nome da ação, insira **PutRecord**.
10. Em Perfil de execução, digite o ARN para o perfil de execução.
11. Deixe o padrão de Passagem para Manuseio de conteúdo.
12. Escolha Criar método.
13. Na seção Solicitação de integração, adicione os seguintes parâmetros de cabeçalhos de solicitação de URL:

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

A tarefa segue o mesmo procedimento para configurar o mapeamento do parâmetro de solicitação para o método GET `/streams`.

14. Adicione o seguinte modelo de mapeamento de corpo para mapear dados na solicitação do método PUT `/streams/{stream-name}/record` à solicitação de integração correspondente de POST `/?Action=PutRecord`:

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

Esse modelo de mapeamento pressupõe que a carga da solicitação de método seja do seguinte formato:

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


Esses dados podem ser modelados pelo seguinte 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" }
  }
}
```

Você pode criar um modelo para incluir esse esquema e usar o modelo para facilitar a geração do modelo de mapeamento. No entanto, pode gerar um modelo de mapeamento sem usar qualquer modelo.

15. Para testar o método **PUT /streams/{stream-name}/record**, defina a variável de caminho **stream-name** como o nome de um fluxo existente, forneça uma carga do formato necessário e, em seguida, envie a solicitação de método. O resultado bem-sucedido é uma resposta **200 OK** com uma carga no seguinte formato:

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

Como configurar e testar o método **PUT /streams/{stream-name}/records** para invocar **PutRecords** no Kinesis

1. Selecione o recurso **/records** e **Criar método**.
2. Em **Tipo de método**, selecione **PUT**.
3. Em **Tipo de integração**, selecione **Serviço da AWS**.
4. Em **Região da AWS**, selecione a Região da AWS onde você criou o fluxo do Kinesis.
5. Em **AWS service (Serviço da AWS)**, selecione **Kinesis**.
6. Mantenha o subdomínio da AWS em branco.

7. Em Método HTTP, escolha POST.
8. Em Tipo de ação, selecione Usar nome da ação.
9. Em Nome da ação, insira **PutRecords**.
10. Em Perfil de execução, digite o ARN para o perfil de execução.
11. Deixe o padrão de Passagem para Manuseio de conteúdo.
12. Escolha Criar método.
13. Na seção Solicitação de integração, adicione os seguintes parâmetros de cabeçalhos de solicitação de URL:

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

A tarefa segue o mesmo procedimento para configurar o mapeamento do parâmetro de solicitação para o método GET /streams.

14. Adicione o seguinte modelo de mapeamento para associar dados na solicitação do método PUT /streams/{stream-name}/records à solicitação de integração correspondente de POST /?Action=PutRecords:

```
{
  "StreamName": "$input.params('stream-name')",
  "Records": [
    #foreach($elem in $input.path('$.records'))
    {
      "Data": "$util.base64Encode($elem.data)",
      "PartitionKey": "$elem.partition-key"
    }#if($foreach.hasNext),#end
  ]#end
}
```

Esse modelo de mapeamento pressupõe que a carga da solicitação do método pode ser modelada pelo seguinte esquema JSON:

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

Você pode criar um modelo para incluir esse esquema e usar o modelo para facilitar a geração do modelo de mapeamento. No entanto, pode gerar um modelo de mapeamento sem usar qualquer modelo.

Neste tutorial, usamos dois formatos de carga um pouco diferentes para ilustrar que um desenvolvedor de API pode optar por expor o formato de dados de backend ao cliente ou ocultá-lo do cliente. Um formato é para o método PUT `/streams/{stream-name}/records` (acima). O outro formato é usado para o método PUT `/streams/{stream-name}/record` (no procedimento anterior). Em um ambiente de produção, você deve manter os dois formatos consistentes.

15.

Para testar o método PUT `/streams/{stream-name}/records`, defina a variável de caminho `stream-name` como um fluxo existente, forneça a carga a seguir e envie a solicitação de método.

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

O resultado bem-sucedido é uma resposta 200 OK com uma carga semelhante à saída a seguir:

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

Como configurar e testar o método **GET /streams/{stream-name}/sharditerator**, invoque **GetShardIterator** no Kinesis

O método **GET /streams/{stream-name}/sharditerator** é um método auxiliar para adquirir um iterador de fragmentos necessário antes de chamar o método **GET /streams/{stream-name}/records**.

1. Selecione o recurso /sharditerator e Criar método.
2. Em Tipo de método, selecione GET.
3. Em Tipo de integração, selecione Serviço da AWS.
4. Em Região da AWS, selecione a Região da AWS onde você criou o fluxo do Kinesis.
5. Em AWS service (Serviço da AWS), selecione Kinesis.
6. Mantenha o subdomínio da AWS em branco.
7. Em Método HTTP, escolha POST.
8. Em Tipo de ação, selecione Usar nome da ação.
9. Em Nome da ação, insira **GetShardIterator**.
10. Em Perfil de execução, digite o ARN para o perfil de execução.
11. Deixe o padrão de Passagem para Manuseio de conteúdo.

12. Selecione Parâmetros de string de consulta de URL.

A ação `GetShardIterator` requer uma entrada de um valor `ShardId`. Para transmitir um valor `ShardId` fornecido pelo cliente, adicionamos um parâmetro de consulta `shard-id` à solicitação de método, conforme mostrado na etapa a seguir.

13. Escolha Add query string (Adicionar string de consulta).

14. Em Nome, digite **shard-id**.

15. Mantenha Obrigatório e Armazenamento em cache desativados.

16. Escolha Criar método.

17. Na seção Solicitação de integração, adicione o modelo de mapeamento a seguir para gerar a entrada necessária (`ShardId` e `StreamName`) à ação `GetShardIterator` dos parâmetros `shard-id` e `stream-name` da solicitação de método. Além disso, o modelo de mapeamento também define `ShardIteratorType` como `TRIM_HORIZON` como um padrão.

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

18. Usando a opção Test (Testar) no console do API Gateway, insira um nome de stream existente como o valor da variável `stream-name` Path (Caminho), defina `shard-id` Query string (String de consulta) como um valor `ShardId` existente (por exemplo, `shard-0000000000004`), e escolha Test (Testar).

A carga da resposta bem-sucedida é semelhante à saída a seguir:

```
{
  "ShardIterator": "AAAAAAAAAFYVN3V1Fy..."
}
```

Anote o valor de `ShardIterator`. Você precisa dela para obter registros de um fluxo.

Como configurar e testar o método **GET /streams/{stream-name}/records** para invocar a ação **GetRecords** no Kinesis

1. Selecione o recurso /records e Criar método.

2. Em Tipo de método, selecione GET.
3. Em Tipo de integração, selecione Serviço da AWS.
4. Em Região da AWS, selecione a Região da AWS onde você criou o fluxo do Kinesis.
5. Em AWS service (Serviço da AWS), selecione Kinesis.
6. Mantenha o subdomínio da AWS em branco.
7. Em Método HTTP, escolha POST.
8. Em Tipo de ação, selecione Usar nome da ação.
9. Em Nome da ação, insira **GetRecords**.
10. Em Perfil de execução, digite o ARN para o perfil de execução.
11. Deixe o padrão de Passagem para Manuseio de conteúdo.
12. Escolha Cabeçalhos de solicitação HTTP.

A ação `GetRecords` requer uma entrada de um valor `ShardIterator`. Para transmitir um valor `ShardIterator` fornecido pelo cliente, adicionamos um parâmetro de cabeçalho `Shard-Iterator` à solicitação de método.

13. Escolha Add header (Adicionar cabeçalho).
14. Em Nome, digite **Shard-Iterator**.
15. Mantenha Obrigatório e Armazenamento em cache desativados.
16. Escolha Criar método.
17. Na seção Solicitação de integração, adicione o modelo de mapeamento do corpo a seguir para associar o valor do parâmetro de cabeçalho `Shard-Iterator` ao valor da propriedade `ShardIterator` da carga útil de JSON para a ação `GetRecords` no Kinesis.

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

18. Usando a opção Testar no console do API Gateway, digite um nome de fluxo existente como o valor da variável Caminho de stream-name, defina o Cabeçalho de `Shard-Iterator` como o valor `ShardIterator` obtido da execução de teste do método GET `/streams/{stream-name}/sharditerator` (acima) e selecione Testar.

A carga da resposta bem-sucedida é semelhante à saída a seguir:

```
{
```

```
"MillisBehindLatest": 0,  
"NextShardIterator": "AAAAAAAAAAAF...",  
"Records": [ ... ]  
}
```

Definições do OpenAPI de uma API demonstrativa como um proxy do Kinesis

Veja a seguir as definições do OpenAPI para a API demonstrativa como um proxy do Kinesis usado neste tutorial.

OpenAPI 3.0

```
{  
  "openapi": "3.0.0",  
  "info": {  
    "title": "KinesisProxy",  
    "version": "2016-03-31T18:25:32Z"  
  },  
  "paths": {  
    "/streams/{stream-name}/sharditerator": {  
      "get": {  
        "parameters": [  
          {  
            "name": "stream-name",  
            "in": "path",  
            "required": true,  
            "schema": {  
              "type": "string"  
            }  
          },  
          {  
            "name": "shard-id",  
            "in": "query",  
            "schema": {  
              "type": "string"  
            }  
          }  
        ],  
        "responses": {  
          "200": {  
            "description": "200 response",  
            "content": {
```

```

        "application/json": {
            "schema": {
                "$ref": "#/components/schemas/Empty"
            }
        }
    },
    "x-amazon-apigateway-integration": {
        "type": "aws",
        "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
        "uri": "arn:aws:apigateway:us-east-1:kinesis:action/GetShardIterator",
        "responses": {
            "default": {
                "statusCode": "200"
            }
        },
        "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
        },
        "requestTemplates": {
            "application/json": "{$\n    \"ShardId\": \"${input.params('shard-
id')}\",\n    \"ShardIteratorType\": \"TRIM_HORIZON\",\n    \"StreamName\":
\"${input.params('stream-name')}\",\n}"
        },
        "passthroughBehavior": "when_no_match",
        "httpMethod": "POST"
    }
},
"/streams/{stream-name}/records": {
    "get": {
        "parameters": [
            {
                "name": "stream-name",
                "in": "path",
                "required": true,
                "schema": {
                    "type": "string"
                }
            },
            {
                "name": "Shard-Iterator",

```



```

        "in": "header",
        "schema": {
            "type": "string"
        }
    },
    ],
    "responses": {
        "200": {
            "description": "200 response",
            "content": {
                "application/json": {
                    "schema": {
                        "$ref": "#/components/schemas/Empty"
                    }
                }
            }
        }
    },
    },
    "x-amazon-apigateway-integration": {
        "type": "aws",
        "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
        "uri": "arn:aws:apigateway:us-east-1:kinesis:action/GetRecords",
        "responses": {
            "default": {
                "statusCode": "200"
            }
        },
        "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
        },
        "requestTemplates": {
            "application/json": "{\n    \"ShardIterator\": \"\${input.params('Shard-
Iterator')}\",\n}"
        },
        "passthroughBehavior": "when_no_match",
        "httpMethod": "POST"
    }
},
"put": {
    "parameters": [
        {
            "name": "Content-Type",
            "in": "header",

```

```
        "schema": {
          "type": "string"
        }
      },
      {
        "name": "stream-name",
        "in": "path",
        "required": true,
        "schema": {
          "type": "string"
        }
      }
    ],
    "requestBody": {
      "content": {
        "application/json": {
          "schema": {
            "$ref": "#/components/schemas/PutRecordsMethodRequestPayload"
          }
        },
        "application/x-amz-json-1.1": {
          "schema": {
            "$ref": "#/components/schemas/PutRecordsMethodRequestPayload"
          }
        }
      },
      "required": true
    },
    "responses": {
      "200": {
        "description": "200 response",
        "content": {
          "application/json": {
            "schema": {
              "$ref": "#/components/schemas/Empty"
            }
          }
        }
      }
    }
  },
  "x-amazon-apigateway-integration": {
    "type": "aws",
    "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
    "uri": "arn:aws:apigateway:us-east-1:kinesis:action/PutRecords",
```

```

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

```

```

        }
    }
}
},
"x-amazon-apigateway-integration": {
    "type": "aws",
    "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
    "uri": "arn:aws:apigateway:us-east-1:kinesis:action/DescribeStream",
    "responses": {
        "default": {
            "statusCode": "200"
        }
    },
    "requestTemplates": {
        "application/json": "{\n    \"StreamName\": \"${input.params('stream-
name')}\n}"
    },
    "passthroughBehavior": "when_no_match",
    "httpMethod": "POST"
}
},
"post": {
    "parameters": [
        {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "schema": {
                "type": "string"
            }
        }
    ],
    "responses": {
        "200": {
            "description": "200 response",
            "content": {
                "application/json": {
                    "schema": {
                        "$ref": "#/components/schemas/Empty"
                    }
                }
            }
        }
    }
}
},
},
},

```

```

"x-amazon-apigateway-integration": {
  "type": "aws",
  "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
  "uri": "arn:aws:apigateway:us-east-1:kinesis:action/CreateStream",
  "responses": {
    "default": {
      "statusCode": "200"
    }
  },
  "requestParameters": {
    "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
  },
  "requestTemplates": {
    "application/json": "{\n  \"ShardCount\": 5,\n  \"StreamName\":
\"$input.params('stream-name')\"\n}"
  },
  "passthroughBehavior": "when_no_match",
  "httpMethod": "POST"
},
"delete": {
  "parameters": [
    {
      "name": "stream-name",
      "in": "path",
      "required": true,
      "schema": {
        "type": "string"
      }
    }
  ],
  "responses": {
    "200": {
      "description": "200 response",
      "headers": {
        "Content-Type": {
          "schema": {
            "type": "string"
          }
        }
      }
    },
    "content": {
      "application/json": {

```

```

        "schema": {
            "$ref": "#/components/schemas/Empty"
        }
    },
    },
    "400": {
        "description": "400 response",
        "headers": {
            "Content-Type": {
                "schema": {
                    "type": "string"
                }
            }
        },
        "content": {}
    },
    "500": {
        "description": "500 response",
        "headers": {
            "Content-Type": {
                "schema": {
                    "type": "string"
                }
            }
        },
        "content": {}
    },
    },
    "x-amazon-apigateway-integration": {
        "type": "aws",
        "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
        "uri": "arn:aws:apigateway:us-east-1:kinesis:action/DeleteStream",
        "responses": {
            "4\\d{2}": {
                "statusCode": "400",
                "responseParameters": {
                    "method.response.header.Content-Type":
"integration.response.header.Content-Type"
                }
            },
            "default": {
                "statusCode": "200",
                "responseParameters": {

```

```

        "method.response.header.Content-Type":
"integration.response.header.Content-Type"
    }
  },
  "5\\d{2}": {
    "statusCode": "500",
    "responseParameters": {
      "method.response.header.Content-Type":
"integration.response.header.Content-Type"
    }
  }
},
"requestParameters": {
  "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
},
"requestTemplates": {
  "application/json": "{\n    \"StreamName\": \"${input.params('stream-
name')}\n\n}"
},
"passthroughBehavior": "when_no_match",
"httpMethod": "POST"
}
}
},
"/streams/{stream-name}/record": {
  "put": {
    "parameters": [
      {
        "name": "stream-name",
        "in": "path",
        "required": true,
        "schema": {
          "type": "string"
        }
      }
    ]
  },
  "responses": {
    "200": {
      "description": "200 response",
      "content": {
        "application/json": {
          "schema": {
            "$ref": "#/components/schemas/Empty"
          }
        }
      }
    }
  }
}
}

```

```

        }
      }
    }
  },
  "x-amazon-apigateway-integration": {
    "type": "aws",
    "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
    "uri": "arn:aws:apigateway:us-east-1:kinesis:action/PutRecord",
    "responses": {
      "default": {
        "statusCode": "200"
      }
    },
    "requestParameters": {
      "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
    },
    "requestTemplates": {
      "application/json": "{\n  \"StreamName\": \"${input.params('stream-
name')}\",\n  \"Data\": \"${util.base64Encode($input.json('$.Data'))}\",\n
  \"PartitionKey\": \"${input.path('$.PartitionKey')}\",\n}"
    },
    "passthroughBehavior": "when_no_match",
    "httpMethod": "POST"
  }
},
"/streams": {
  "get": {
    "responses": {
      "200": {
        "description": "200 response",
        "content": {
          "application/json": {
            "schema": {
              "$ref": "#/components/schemas/Empty"
            }
          }
        }
      }
    }
  },
  "x-amazon-apigateway-integration": {
    "type": "aws",

```



```

        "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
        "uri": "arn:aws:apigateway:us-east-1:kinesis:action/ListStreams",
        "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\n}"
    },
    "passthroughBehavior": "when_no_match",
    "httpMethod": "POST"
  }
},
"post": {
  "consumes": [
    "application/json"
  ],
  "produces": [
    "application/json"
  ],
  "parameters": [
    {
      "name": "stream-name",
      "in": "path",
      "required": true,
      "type": "string"
    }
  ],
  "responses": {
    "200": {
      "description": "200 response",
      "schema": {
        "$ref": "#/definitions/Empty"
      }
    }
  },
  "x-amazon-apigateway-integration": {
    "type": "aws",
    "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
    "uri": "arn:aws:apigateway:us-east-1:kinesis:action/CreateStream",
    "responses": {
      "default": {
        "statusCode": "200"
      }
    },
    "requestParameters": {
      "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
    },
  },

```

```

    "requestTemplates": {
      "application/json": "{\n    \"ShardCount\": 5,\n    \"StreamName\":\n    \"${input.params('stream-name')}\n    }\n    },\n    \"passthroughBehavior\": \"when_no_match\",\n    \"httpMethod\": \"POST\"\n  }\n},\n\"delete\": {\n  \"consumes\": [\n    \"application/json\"\n  ],\n  \"produces\": [\n    \"application/json\"\n  ],\n  \"parameters\": [\n    {\n      \"name\": \"stream-name\",\n      \"in\": \"path\",\n      \"required\": true,\n      \"type\": \"string\"\n    }\n  ],\n  \"responses\": {\n    \"200\": {\n      \"description\": \"200 response\",\n      \"schema\": {\n        \"$ref\": \"#/definitions/Empty\"\n      },\n      \"headers\": {\n        \"Content-Type\": {\n          \"type\": \"string\"\n        }\n      }\n    },\n    \"400\": {\n      \"description\": \"400 response\",\n      \"headers\": {\n        \"Content-Type\": {\n          \"type\": \"string\"\n        }\n      }\n    },\n    \"500\": {\n
```

```

        "description": "500 response",
        "headers": {
            "Content-Type": {
                "type": "string"
            }
        }
    },
    "x-amazon-apigateway-integration": {
        "type": "aws",
        "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
        "uri": "arn:aws:apigateway:us-east-1:kinesis:action/DeleteStream",
        "responses": {
            "4\\d{2}": {
                "statusCode": "400",
                "responseParameters": {
                    "method.response.header.Content-Type":
"integration.response.header.Content-Type"
                }
            },
            "default": {
                "statusCode": "200",
                "responseParameters": {
                    "method.response.header.Content-Type":
"integration.response.header.Content-Type"
                }
            },
            "5\\d{2}": {
                "statusCode": "500",
                "responseParameters": {
                    "method.response.header.Content-Type":
"integration.response.header.Content-Type"
                }
            }
        },
        "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
        },
        "requestTemplates": {
            "application/json": "{\n    \"StreamName\": \"$input.params('stream-
name')\"\n}"
        },
        "passthroughBehavior": "when_no_match",

```

```

        "httpMethod": "POST"
      }
    },
    "/streams/{stream-name}/record": {
      "put": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          }
        }
      },
      "x-amazon-apigateway-integration": {
        "type": "aws",
        "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
        "uri": "arn:aws:apigateway:us-east-1:kinesis:action/PutRecord",
        "responses": {
          "default": {
            "statusCode": "200"
          }
        },
        "requestParameters": {
          "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
        },
        "requestTemplates": {

```

```

        "application/json": "{\n    \"StreamName\": \"\${input.params('stream-
name')}\",\n    \"Data\": \"\${util.base64Encode($input.json('$.Data'))}\",\n
    \"PartitionKey\": \"\${input.path('$.PartitionKey')}\",\n}"
    },
    "passthroughBehavior": "when_no_match",
    "httpMethod": "POST"
  }
},
"/streams/{stream-name}/records": {
  "get": {
    "consumes": [
      "application/json"
    ],
    "produces": [
      "application/json"
    ],
    "parameters": [
      {
        "name": "stream-name",
        "in": "path",
        "required": true,
        "type": "string"
      },
      {
        "name": "Shard-Iterator",
        "in": "header",
        "required": false,
        "type": "string"
      }
    ],
    "responses": {
      "200": {
        "description": "200 response",
        "schema": {
          "$ref": "#/definitions/Empty"
        }
      }
    }
  },
  "x-amazon-apigateway-integration": {
    "type": "aws",
    "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
    "uri": "arn:aws:apigateway:us-east-1:kinesis:action/GetRecords",
    "responses": {

```



```

        "default": {
            "statusCode": "200"
        },
        "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
        },
        "requestTemplates": {
            "application/json": "{\n    \"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  \n  \"StreamName\": \"$input.params('stream-
name')\",\n  \n  \"Records\": [\n    {\n      \n      \"Data\":\n        \"$util.base64Encode($elem.data)\",\n      \n      \"PartitionKey\":\n        \"$elem.partition-key\"\n    }#if($foreach.hasNext),#end\n  ]\n}",
      "application/x-amz-json-1.1": "{\n  \n  \"StreamName\":\n    \"$input.params('stream-name')\",\n  \n  \"records\" : [\n    {\n      \n      \"Data\n    \" : \"$elem.data\",\n      \n      \"PartitionKey\" : \"$elem.partition-key\"\n    }\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": "{$input.params('shard-id')}\n\n\"ShardId\": \"\n\n\"ShardIteratorType\": \"TRIM_HORIZON\", \n\n\"StreamName\": \"\n\n\"$input.params('stream-name')\"\n\n}"
    },
    "passthroughBehavior": "when_no_match",
    "httpMethod": "POST"
  }
}
},
"definitions": {
  "Empty": {
    "type": "object"
  },
  "PutRecordsMethodRequestPayload": {
    "type": "object",
    "properties": {
      "records": {
        "type": "array",
        "items": {
          "type": "object",
          "properties": {
            "data": {
              "type": "string"
            },
            "partition-key": {
              "type": "string"
            }
          }
        }
      }
    }
  }
}
```

Tutorial: como criar uma API REST usando SDKs da AWS ou a AWS CLI

Use o tutorial a seguir para criar uma API PetStore compatível com os métodos GET /pets e GET /pets/{petId}. Os dois métodos são integrados ao endpoint HTTP. É possível seguir este tutorial

usando o AWS SDK para JavaScript, o SDK para Python (Boto3) ou a AWS CLI. Use as seguintes funções ou comandos para configurar sua 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 saber mais sobre o AWS SDK para JavaScript v3, consulte [What's the AWS SDK for JavaScript?](#). Para saber mais sobre o SDK for Python (Boto3), consulte [AWS SDK para Python \(Boto3\)](#). Para obter mais informações sobre a AWS CLI, consulte [O que é a AWS CLI?](#).

Configurar uma API PetStore otimizada para bordas

Neste tutorial, os exemplos de comandos usam valores de espaço reservado para IDs de valor, como ID de API e ID de recurso. Durante a conclusão do tutorial, substitua os valores de exemplo por seus próprios valores.

Como configurar uma API PetStore otimizada para bordas usando AWS SDKs

1. Use o seguinte exemplo para criar uma entidade 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)
}
})();
```

Uma chamada bem-sucedida retorna o ID da API e o ID do recurso-raiz da API em uma saída como a seguinte:

```
{
  id: 'abc1234',
  name: 'PetStore (JavaScript v3 SDK)',
  description: 'Demo API created using the AWS SDK for node.js',
```

```

    createdAt: 2017-09-05T19:32:35.000Z,
    version: '0.00.001',
    rootResourceId: 'efg567'
    binaryMediaTypes: [ '*' ]
  }

```

Python

```

import boto3
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.create_rest_api(
        name='Simple PetStore (Python SDK)',
        description='Demo API created using the AWS SDK for Python',
        version='0.00.001',
        binaryMediaTypes=[
            '*'
        ]
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Couldn't create REST API %s.", error)
    raise
attribute=["id","name","description","createdAt","version","binaryMediaTypes","apiKeySource"]
filtered_result = {key:result[key] for key in attribute}
print(filtered_result)

```

Uma chamada bem-sucedida retorna o ID da API e o ID do recurso-raiz da API em uma saída como a seguinte:

```

{'id': 'abc1234', 'name': 'Simple PetStore (Python SDK)', 'description':
'Demo API created using the AWS SDK for Python', 'createdAt':
datetime.datetime(2024, 4, 3, 14, 31, 39, tzinfo=tzlocal()), 'version':
'0.00.001', 'binaryMediaTypes': ['*'], 'apiKeySource': 'HEADER',
'endpointConfiguration': {'types': ['EDGE']}, 'disableExecuteApiEndpoint':
False, 'rootResourceId': 'efg567'}

```

AWS CLI

```
aws apigateway create-rest-api --name 'Simple PetStore (AWS CLI)' --region us-west-2
```

Veja a seguir a saída desse 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"
}
```

A API criada tem um ID de API abcd1234 e um ID de recurso-raiz de efg567. Você usa esses valores na configuração da sua API.

- Depois, anexe um recurso filho sob a raiz e especifique o `RootResourceId` como o valor da propriedade `parentId`. Use o seguinte exemplo para criar um recurso `/pets` para a 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)  
}  
})();
```

Uma chamada bem-sucedida retorna informações sobre seu recurso em uma saída como a seguinte:

```
{  
  "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)
```

Uma chamada bem-sucedida retorna informações sobre seu recurso em uma saída como a seguinte:

```
{'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
```

Veja a seguir a saída desse comando:

```
{  
  "id": "aaa111",  
  "parentId": "efg567",  
  "pathPart": "pets",  
  "path": "/pets"  
}
```

O recurso `/pets` criado tem o ID de recurso de `aaa111`. Você usa esse valor na configuração da sua API.

3. Depois, anexe um recurso filho sobre o recurso `/pets`. Esse recurso `/petId` tem um parâmetro de caminho para `{petId}`. Para incluir um caminho em um parâmetro de caminho, coloque-o entre chaves, `{ }`. Use o seguinte exemplo para criar um recurso `/pets/{petId}` para a 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)  
  }  
})
```

```
}  
})();
```

Uma chamada bem-sucedida retorna informações sobre seu recurso em uma saída como a seguinte:

```
{  
  "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)
```

Uma chamada bem-sucedida retorna informações sobre seu recurso em uma saída como a seguinte:

```
{'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}'
```

Veja a seguir a saída desse comando:

```
{  
  "id": "bbb222",  
  "parentId": "aaa111",  
  "path": "/pets/{petId}",  
  "pathPart": "{petId}"  
}
```

O recurso `/pets/{petId}` criado tem o ID de recurso de `bbb222`. Você usa esse valor na configuração da sua API.

4. Nas duas etapas a seguir, adicione métodos HTTP aos recursos. Neste tutorial, você definirá os métodos para ter acesso aberto, configurando `authorization-type` como `NONE`. Para permitir que somente usuários autenticados chamem o método, você pode usar as funções e políticas do IAM, um autorizador do Lambda (anteriormente conhecido como autorizador personalizado) ou um grupo de usuários do Amazon Cognito. Para obter mais informações, consulte [the section called “Controle de acesso”](#).

Use o seguinte exemplo para adicionar o método HTTP GET ao 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)
}
})();
```

Uma chamada bem-sucedida retorna a seguinte saída:

```
{
  "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)
```

Uma chamada bem-sucedida retorna a seguinte saída:

```
{'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
```

Veja a seguir a saída desse comando:

```
{  
  "httpMethod": "GET",  
  "authorizationType": "NONE",  
  "apiKeyRequired": false  
}
```

5. Use o seguinte exemplo para adicionar o método HTTP GET ao recurso `/pets/{petId}`, definir a propriedade `requestParameters` e passar o valor `petId` fornecido pelo cliente para o 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)
    }
    })();
```

Uma chamada bem-sucedida retorna a seguinte saída:

```
{
  "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)
```

Uma chamada bem-sucedida retorna a seguinte saída:

```
{'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
```

Veja a seguir a saída desse comando:

```
{  
  "httpMethod": "GET",  
  "authorizationType": "NONE",  
  "apiKeyRequired": false,  
  "requestParameters": {  
    "method.request.path.petId": true  
  }  
}
```

6. Use o seguinte exemplo para adicionar a resposta de método 200 OK para o 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)
    }
}
})();
```

Uma chamada bem-sucedida retorna a seguinte saída:

```
{
  "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)
```

Uma chamada bem-sucedida retorna a seguinte saída:

```
{'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
```

Veja a seguir a saída desse comando:

```
{  
  "statusCode": "200"  
}
```

7. Use o seguinte exemplo para adicionar a resposta de método 200 OK para o 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)  
  }  
})();
```

Uma chamada bem-sucedida retorna a seguinte saída:

```
{  
  "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)
```

Uma chamada bem-sucedida retorna a seguinte saída:

```
{'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
```

Veja a seguir a saída desse comando:

```
{
  "statusCode": "200"
}
```

8. Use o exemplo a seguir para configurar uma integração para o método GET /pets com um endpoint HTTP. O endpoint HTTPS é `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)
}
})();
```

Uma chamada bem-sucedida retorna a seguinte saída:

```
{
  "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)
```

Uma chamada bem-sucedida retorna a seguinte saída:

```
{'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
```

Veja a seguir a saída desse 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 o exemplo a seguir para configurar uma integração para o método GET `/pets/{petId}` com um endpoint HTTP. O endpoint HTTPS é `http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}`. Nesta etapa, associe o parâmetro de caminho `petId` ao parâmetro de caminho no endpoint de integração de `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)
  }
})();

```

Uma chamada bem-sucedida retorna a seguinte saída:

```
{
```

```
"httpMethod": "GET",
"passthroughBehavior": "WHEN_NO_MATCH",
"cacheKeyParameters": [],
"type": "HTTP",
"uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}",
"cacheNamespace": "ddd444",
"requestParameters": {
    "integration.request.path.id": "method.request.path.petId"
}
}
```

Python

```
import boto3
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.put_integration(
        restApiId='ieps9b05sf',
        resourceId='t8zeb4',
        httpMethod='GET',
        type='HTTP',
        integrationHttpMethod='GET',
        uri='http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}',
        requestParameters={
            "integration.request.path.id": "method.request.path.petId"
        }
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Set up the integration of the 'GET /pets/{petId}' method of the API failed %s.", error)
    raise

attribute=["httpMethod","passthroughBehavior","cacheKeyParameters", "type",
"uri", "cacheNamespace", "requestParameters"]
filtered_result = {key:result[key] for key in attribute}
print(filtered_result)
```

Uma chamada bem-sucedida retorna a seguinte saída:

```
{'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
```

Veja a seguir a saída desse 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 o seguinte exemplo para adicionar a resposta da integração para a integração 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)
  }
})();
```

Uma chamada bem-sucedida retorna a seguinte saída:

```
{
  "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)
```

Uma chamada bem-sucedida retorna a seguinte saída:

```
{'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
```

Veja a seguir a saída desse comando:

```
{
  "statusCode": "200",
  "selectionPattern": ""
}
```

11. Use o seguinte exemplo para adicionar a resposta da integração para a integração 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)
    }
})();
```

Uma chamada bem-sucedida retorna a seguinte saída:

```
{
  "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)
```

Uma chamada bem-sucedida retorna a seguinte saída:

```
{'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
```

Veja a seguir a saída desse comando:

```
{  
  "statusCode": "200",  
  "selectionPattern": ""  
}
```

Depois que você criar a resposta de integração, sua API poderá consultar os animais de estimação disponíveis no site da PetStore e visualizar um animal de estimação individual de um identificador especificado. Antes que sua API possa ser chamada pelos clientes, você deve implantá-la. Recomendamos que você teste a API antes de implantá-la.

12. Use o seguinte exemplo para testar o 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 o seguinte exemplo para testar o método GET /pets/{petId} usando 3 como petId:

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'
```

Depois de testar a API com sucesso, implante-a em um estágio.

14. Use o exemplo a seguir para implantar a API em um estágio chamado `test`. Quando você implanta a API em um estágio, os chamadores da API podem invocá-la.

JavaScript v3

```
import {APIGatewayClient, CreateDeploymentCommand } from "@aws-sdk/client-api-gateway";
(async function (){
const apig = new APIGatewayClient({region:"us-east-1"});
const command = new CreateDeploymentCommand({
  restApiId: 'abcd1234',
  stageName: 'test',
  stageDescription: 'test deployment'
});
try {
  const results = await apig.send(command)
  console.log("Deploying API succeeded\n", results)
} catch (err) {
  console.log("Deploying API failed:\n", err)
}
})();
```

Python

```
import boto3
import logging

logger = logging.getLogger()
apig = boto3.client('apigateway')

try:
    result = apig.create_deployment(
        restApiId='ieps9b05sf',
        stageName='test',
        stageDescription='my test stage',
    )
except botocore.exceptions.ClientError as error:
    logger.exception("Error deploying stage %s.", error)
    raise
print('Deploying API succeeded')
print(result)
```

AWS CLI

```
aws apigateway create-deployment --rest-api-id abcd1234 \  
  --region us-west-2 \  
  --stage-name test \  
  --stage-description 'Test stage' \  
  --description 'First deployment'
```

Veja a seguir a saída desse comando:

```
{  
  "id": "ab1c1d",  
  "description": "First deployment",  
  "createdDate": "2022-12-15T08:44:13-08:00"  
}
```

Sua API agora pode ser chamada pelos clientes. Você pode testar essa API inserindo o URL `https://abcd1234.execute-api.us-west-2.amazonaws.com/test/pets` em um navegador e substituindo `abcd1234` pelo identificador da sua API.

Para ver mais exemplos de como criar ou atualizar uma API usando AWS SDKs ou a AWS CLI, consulte [Actions for API Gateway using AWS SDKs](#).

Automatizar a configuração da API

Em vez de criar a API passo a passo, você pode automatizar a criação e a limpeza de recursos da AWS usando a OpenAPI, o CloudFormation ou o Terraform para criar a API.

Definição da OpenAPI 3.0

Você pode importar uma definição da OpenAPI para o API Gateway. Para obter mais informações, 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" : { }
}

```

Modelo do AWS CloudFormation

Para implantar um modelo do CloudFormation, consulte [Criar uma pilha no console do 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'

```

Configuração do Terraform

Para obter mais informações sobre o 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: Crie uma API REST privada

Neste tutorial, você cria uma API REST privada. Os clientes podem acessar a API somente de dentro de sua Amazon VPC. A API é isolada da internet pública, que é um requisito de segurança comum.

Este tutorial leva aproximadamente 30 minutos para ser concluído. Primeiro, você usa um modelo do CloudFormation para criar uma Amazon VPC, um endpoint da VPC, uma função do AWS Lambda e executar uma instância do Amazon EC2 que você usará para testar sua API. Em seguida, use o Console de gerenciamento da AWS para criar uma API privada e anexar uma política de recursos que permita o acesso somente a partir do endpoint da VPC. Por fim, você testa a sua API.



Para concluir esse tutorial, você precisa de uma conta da AWS e de um usuário do AWS Identity and Access Management com acesso ao console. Para obter mais informações, consulte [Pré-requisitos](#).

Neste tutorial, você usará o Console de gerenciamento da AWS. Para obter um modelo do CloudFormation que cria essa API e todos os recursos relacionados, consulte [template.yaml](#).

Tópicos

- [Etapa 1: Criar dependências](#)
- [Etapa 2: Crie uma API privada](#)
- [Etapa 3: Crie um método e integração](#)
- [Etapa 4: Anexe uma política de recursos](#)
- [Etapa 5: Implante sua API](#)
- [Etapa 6: Verifique se sua API não está acessível publicamente](#)
- [Etapa 7: Conecte-se a uma instância em sua VPC e invoque sua API](#)
- [Etapa 8: Limpar](#)
- [Próximas etapas: Automatize com CloudFormation](#)

Etapa 1: Criar dependências

Baixe e descompacte [este modelo do CloudFormation](#). Você usa o modelo para criar todas as dependências para sua API privada, incluindo uma Amazon VPC, um endpoint da VPC e uma função do Lambda que serve como backend de sua API. Você cria a API privada mais tarde.

Como criar uma pilha do CloudFormation

1. Abra o console do CloudFormation em <https://console.aws.amazon.com/cloudformation>.
2. Selecione Create stack (Criar pilha) e With new resources (standard) (Com novos recursos (padrão)).
3. Em Specify template (Especificar modelo), escolha Upload a template file (Fazer upload de um arquivo de modelo).
4. Selecione o modelo que você baixou.
5. Escolha Next (Próximo).
6. Em Nome da pilha, insira **private-api-tutorial** e escolha Avançar.
7. Para Configurar opções de pilha, escolha Avançar.

8. Para Capabilities (Recursos), reconheça que CloudFormation pode criar recursos do IAM em sua conta.
9. Escolha Próximo e, depois, Enviar.

CloudFormationO provisiona as dependências da sua API, o que pode levar alguns minutos. Quando o status da sua pilha do CloudFormation for CREATE_COMPLETE, escolha Outputs (Saídas). Observe o ID de endpoint da VPC. Você precisa dele para as etapas posteriores neste tutorial.

Etapa 2: Crie uma API privada

Crie uma API privada para permitir que somente clientes dentro de sua VPC a acessem.

Para criar uma API privada

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha Criar API, em seguida, para API REST, escolha Criar.
3. Em API name (Nome da API), insira **private-api-tutorial**.
4. Em Tipo de endpoint de API, escolha Privado.
5. Em IDs de endpoint da VPC, insira o ID de endpoint da VPC em Saídas da pilha do CloudFormation.
6. Em Tipo de endereço IP, escolha Dualstack.
7. Selecione Criar API.

Etapa 3: Crie um método e integração

Crie um método GET e uma integração do Lambda para lidar com solicitações GET para sua API. Quando um cliente invoca sua API, o API Gateway envia a solicitação para a função do Lambda criada na Etapa 1 e, em seguida, retorna uma resposta ao cliente.

Para criar um método e integração

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione a API.
3. Escolha Criar método.
4. Em Tipo de método, selecione GET.
5. Em Tipo de integração, selecione Função do Lambda.

6. Ative Integração de proxy do Lambda. Com uma integração de proxy do Lambda, o API Gateway envia um evento para o Lambda com uma estrutura definida e transforma a resposta da função do Lambda em uma resposta HTTP.
7. Para Lambda function (Função do Lambda), escolha a função que criou com o modelo do CloudFormation na Etapa 1. O nome da função começa com **private-api-tutorial**.
8. Escolha Criar método.

Etapa 4: Anexe uma política de recursos

Anexe uma [política de recursos](#) à sua API, que permite que os clientes invoquem sua API somente por meio do endpoint da VPC. Para restringir ainda mais o acesso à sua API, você também pode configurar uma [política de endpoint da VPC](#) para o endpoint da VPC, mas isso não é necessário para este tutorial.

Para anexar uma política de recursos

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione a API.
3. Selecione Política de recursos e Criar política.
4. Insira a seguinte política. Substitua o *vpceID* pelo seu ID de endpoint da VPC em Outputs (Saídas) da sua pilha do 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. Escolha Save changes.

Etapa 5: Implante sua API

Em seguida, implante sua API para torná-la disponível para clientes em sua Amazon VPC.

Para implantar uma API

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione a API.
3. Escolha Implantar API.
4. Em Estágio, selecione Novo estágio.
5. Em Stage name (Nome do estágio), insira **test**.
6. (Opcional) Em Description (Descrição), insira uma descrição.
7. Escolha Deploy (Implantar).

Agora você está pronto para testar sua API.

Etapa 6: Verifique se sua API não está acessível publicamente

Use `curl` para verificar se você não pode invocar sua API fora da Amazon VPC.

Para testar sua API

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione a API.
3. No painel de navegação principal, selecione Estágios e escolha o estágio testar.
4. Em Detalhes do estágio, escolha o ícone de cópia para copiar o URL de invocação da API. O URL se parece com `https://abcdef123.execute-api.us-west-2.amazonaws.com/`

test. O endpoint da VPC que foi criado na Etapa 1 tem o DNS privado ativado, portanto, você pode usar o URL fornecido para chamar sua API.

5. Use curl para tentar invocar sua API fora da VPC.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/test
```

O curl indica que o endpoint da sua API não pode ser resolvido. Se você receber uma resposta diferente, volte para a Etapa 2 e certifique-se de escolher Privado para o tipo de endpoint da API.

```
curl: (6) Could not resolve host: abcdef123.execute-api.us-west-2.amazonaws.com/  
test
```

Em seguida, conecte-se a uma instância do Amazon EC2 em sua VPC para chamar sua API.

Etapa 7: Conecte-se a uma instância em sua VPC e invoque sua API

Em seguida, teste sua API de dentro da Amazon VPC. Para acessar sua API privada, conecte-se a uma instância do Amazon EC2 em sua VPC e, em seguida, use curl para chamar sua API. Use o Gerenciador de sessões do Systems Manager para se conectar à instância no navegador.

Para testar sua API

1. Abra o console do Amazon EC2 em <https://console.aws.amazon.com/ec2/>.
2. Selecione Instances (Instâncias).
3. Escolha a instância chamada private-api-tutorial que você criou com o modelo do CloudFormation na Etapa 1.
4. Escolha Conectar e escolha Session Manager.
5. Escolha Conectar para executar uma sessão baseada em navegador na instância.
6. Na sessão do Session Manager, use curl para invocar sua API. Invoque sua API porque está usando uma instância na sua Amazon VPC.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/test
```

Verifique se você obtém a resposta Hello from Lambda!.

Session ID: user-

Instance ID: i-

Terminate

```
sh-4.2$ curl https://[redacted].execute-api.us-west-2.amazonaws.com/prod
"Hello from Lambda!"sh-4.2$
```

Você criou com sucesso uma API que é acessível apenas a partir de sua Amazon VPC e, em seguida, verificou se ela funciona.

Etapa 8: Limpar

Para evitar custos desnecessários, exclua os recursos que você criou como parte desse tutorial. As etapas a seguir excluem sua API REST e sua pilha do CloudFormation.

Para excluir uma API REST

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Na página APIs, selecione uma API. Selecione Ações da API, Excluir API e, depois, confirme a escolha.

Para excluir uma pilha do CloudFormation

1. Abra o console do CloudFormation em <https://console.aws.amazon.com/cloudformation>.
2. Selecione sua pilha do CloudFormation.
3. Escolha Excluir e, em seguida, confirme sua escolha.

Próximas etapas: Automatize com CloudFormation

Você pode automatizar a criação e a limpeza de todos os recursos da AWS envolvidos neste tutorial. Para obter um modelo do CloudFormation de exemplo completo, consulte [template.yaml](#).

Tutoriais da API HTTP do Amazon API Gateway

Os tutoriais a seguir fornecem exercícios práticos para ajudar você a saber mais sobre as APIs HTTP do Amazon API Gateway.

Tópicos

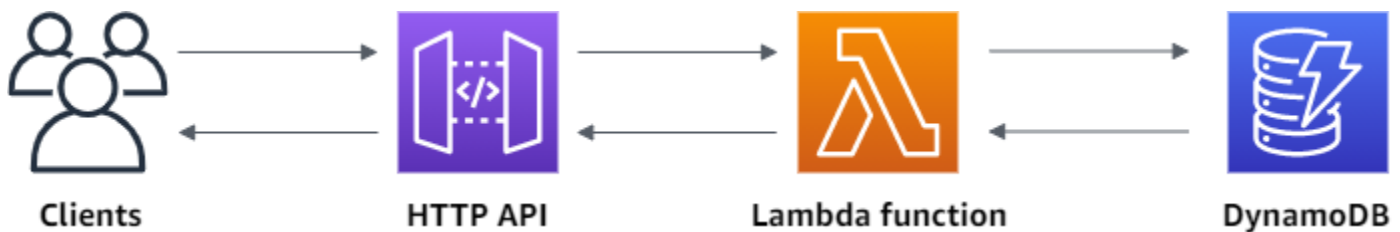
- [Tutorial: Crie uma API HTTP CRUD com o Lambda e o DynamoDB](#)
- [Tutorial: Crie uma API HTTP com uma integração privada a um serviço do Amazon ECS](#)

Tutorial: Crie uma API HTTP CRUD com o Lambda e o DynamoDB

Neste tutorial, você cria uma API sem servidor que cria, lê, atualiza e exclui itens de uma tabela do DynamoDB. O DynamoDB é um serviço de banco de dados NoSQL totalmente gerenciado que proporciona uma performance rápida e previsível com escalabilidade contínua. Este tutorial leva aproximadamente 30 minutos para ser concluído, e você pode fazê-lo no [nível gratuito da AWS](#).

Primeiro, você cria uma tabela do [DynamoDB](#) usando o console do DynamoDB. Em seguida, você cria uma função do [Lambda](#) usando o console do AWS Lambda. Em seguida, crie uma API HTTP usando o console do API Gateway. Por fim, você testa a sua API.

Quando você invoca sua API HTTP, o API Gateway encaminha a solicitação para sua função do Lambda. A função do Lambda interage com o DynamoDB e retorna uma resposta ao API Gateway. O API Gateway retorna uma resposta para você.



Para concluir esse exercício, você precisa de uma conta da AWS e de um usuário do AWS Identity and Access Management com acesso ao console. Para obter mais informações, consulte [Pré-requisitos](#).

Neste tutorial, você usará o Console de gerenciamento da AWS. Para obter um modelo AWS SAM que cria essa API e todos os recursos relacionados, consulte [template.yaml](#).

Tópicos

- [Etapa 1: Crie uma tabela do DynamoDB](#)

- [Etapa 2: Criar uma função do Lambda](#)
- [Etapa 3: Crie uma API HTTP](#)
- [Etapa 4: Crie rotas](#)
- [Etapa 5: Crie uma integração](#)
- [Etapa 6: Anexe a sua integração às rotas](#)
- [Etapa 7: Teste a sua API](#)
- [Etapa 8: Limpar](#)
- [Próximas etapas: Automatizar com AWS SAM ou CloudFormation](#)

Etapa 1: Crie uma tabela do DynamoDB

Você usa uma tabela do [DynamoDB](#) para armazenar dados para a sua API.

Cada item tem um ID exclusivo, que usamos como [partition key](#) (chave de partição) para a tabela.

Como criar uma tabela do DynamoDB

1. Abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.
2. Escolha Create table.
3. Em Table name (Nome da tabela), insira **http-crud-tutorial-items**.
4. Em Partition key, (Chave de partição), insira **id**.
5. Escolha Criar tabela.

Etapa 2: Criar uma função do Lambda

Você cria uma função do [Lambda](#) para o backend da sua API. Essa função do Lambda cria, lê, atualiza e exclui itens do DynamoDB. A função usa [events from API Gateway](#) (eventos do API Gateway) para definir como interagir com o DynamoDB. Para fins de simplicidade, este tutorial usa uma única função do Lambda. Como prática recomendada, você deve criar funções separadas para cada rota. Para obter mais informações, consulte [The Lambda monolith](#) (A aplicação monolítica do Lambda).

Como criar uma função do Lambda

1. Abra o console do Lambda em <https://console.aws.amazon.com/lambda>
2. Escolha Create function (Criar função).

3. Em Function name (Nome da função), insira **http-crud-tutorial-function**.
4. Em Tempo de execução, escolha o último runtime Node.js ou Python compatível.
5. Em Permissions (Permissões), escolha Change default execution role (Alterar a função de execução padrão).
6. Selecione Criar um perfil com base em modelos de política da AWS.
7. Em Role name (Nome da função), insira **http-crud-tutorial-role**.
8. Em Policy templates (Modelos de políticas), escolha **Simple microservice permissions**. Esta política concede à função do Lambda permissão para interagir com o DynamoDB.

 Note

Este tutorial usa uma política gerenciada em prol da simplicidade. Como prática recomendada, você deve criar sua própria política do IAM para conceder as permissões mínimas necessárias.

9. Escolha Create function (Criar função).
10. Abra a função do Lambda no editor de código do console e substitua seu conteúdo pelo código a seguir. Escolha Deploy (Implantar) para atualizar a sua função.

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
```

Etapa 3: Crie uma API HTTP

A API HTTP fornece um endpoint de HTTP para sua função do Lambda. Nesta etapa, você cria uma API vazia. Nas etapas a seguir, você configura rotas e integrações para conectar a sua API e sua função do Lambda.

Para criar uma API HTTP

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha Create API (Criar API) e, em seguida, em API HTTP (API HTTP), escolha Build (Criar).
3. Em API name (Nome da API), insira **http-crud-tutorial-api**.
4. Em Tipo de endereço IP, selecione IPv4.
5. Escolha Próximo.
6. Em Configure routes (Configurar rotas), escolha Next (Próximo) para ignorar a criação da rota. Você cria rotas mais tarde.
7. Analise o estágio que o API Gateway cria para você e escolha Next (Avançar).
8. Escolha Create (Criar).

Etapa 4: Crie rotas

As rotas são uma forma de enviar as solicitações de API recebidas a recursos de backend. As rotas consistem em duas partes: um método HTTP e um caminho de recurso, por exemplo, GET /items. Para esta API de exemplo, criamos quatro rotas:

- GET /items/{id}
- GET /items
- PUT /items
- DELETE /items/{id}

Para criar rotas

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.

2. Selecione a API.
3. Selecione Routes (Rotas).
4. Escolha Create (Criar).
5. Em Método, escolha **GET**.
6. Para o caminho, insira **/items/{id}**. O {id} no final do caminho é um parâmetro de caminho que o API Gateway recupera do caminho de solicitação quando um cliente faz uma solicitação.
7. Escolha Create (Criar).
8. Repita as etapas quatro a sete para GET /itemsDELETE /items/{id} e PUT /items.

Routes Stage: - ▼ Deploy

Routes for http-crud-tutorial-api Create

- ▼ /items
 - GET
 - PUT**
- ▼ /{id}
 - GET
 - DELETE

Route details Delete Edit

PUT /items (ID: bdlb4q)

ARN
 arn:aws:apigateway:us-west-1::apis/abcd1234/routes/aaa111

Authorization
Authorizers protect your API against unauthorized requests. Routes with no authorization attached are open.

No authorizer attached to this route. Attach authorization

Integration
The integration is the backend resource that this route calls when it receives a request.

No integration attached to this route. Attach integration

Etapa 5: Crie uma integração

Você cria uma integração para conectar uma rota aos recursos de backend. Para este exemplo de API, você cria uma integração do Lambda que usa para todas as rotas.

Para criar uma integração

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.

2. Selecione a API.
3. Escolha Integrations (Integrações).
4. Escolha Manage integrations (Gerenciar integrações) e, em seguida, escolha Create (Criar).
5. Pule Attach this integration to a route (Anexar esta integração a uma rota). Você conclui isso em uma etapa posterior.
6. Em Integration type (Tipo de integração), escolha Lambda function (Função do Lambda).
7. Em Lambda function (Função do Lambda), insira **http-crud-tutorial-function**.
8. Escolha Create (Criar).

Etapa 6: Anexe a sua integração às rotas

Para este exemplo de API, você usa a mesma integração do Lambda para todas as rotas. Depois de anexar a integração a todas as rotas da API, sua função do Lambda é invocada quando um cliente chama qualquer uma de suas rotas.

Para anexar integrações a rotas

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione a API.
3. Escolha Integrations (Integrações).
4. Escolha uma rota.
5. Em Choose an existing integration (Escolher uma integração existente), escolha **http-crud-tutorial-function**.
6. Escolha Attach integration (Anexar integração).
7. Repita as etapas quatro a seis para todas as rotas.

Todas as rotas mostram que há uma integração com o AWS Lambda anexada.

API Gateway > APIs > [http-crud-tutorial-api \(abdc1234\)](#) > Integrations

Integrations

Stage: - ▼

Deploy

[Attach integrations to routes](#)

Manage integrations

Routes for http-crud-tutorial-api

Search

▼ /items

GET

AWS Lambda

PUT

AWS Lambda

▼ /{id}

GET

AWS Lambda

DELETE

AWS Lambda

Integration details for route

[Detach integration](#)

[Manage integration](#)

PUT /items (aaa111)

Lambda function

http-crud-tutorial-function (us-west-1) [↗](#)

Integration ID

bbb222

Description

-

Payload format version

The parsing algorithm for the payload sent to and returned from your Lambda function. [Learn more.](#)

2.0 (interpreted response format)

Invoke permissions

The resource policy of the Lambda function determines if API Gateway can invoke it. You can run the AWS CLI command snippet below to give API Gateway permission to invoke your AWS Lambda function.

► Example policy statement

Timeout

The number of milliseconds that API Gateway should wait for a response from the integration before timing out.

30000

Request parameter mapping

Not configured

Response parameter mappings

Not configured

Agora que você tem uma API HTTP com rotas e integrações, já pode testar a sua API.

Etapa 7: Teste a sua API

Para ter certeza de que a sua API está funcionando, use [curl](#).

Para obter a URL para invocar a sua API

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione a API.
3. Observe o URL de invocação da sua API. Ela aparece em Invoke URL (Invocar URL) na página Details (Detalhes).

API Gateway > APIs > http-crud-tutorial-api (abcd1234)

http-crud-tutorial-api

Stage: - ▼

Deploy

API details

Edit

API ID

abcd1234

Protocol

HTTP

Created

2024-05-02

Description

My HTTP CRUD tutorial API

Default endpoint

✔ Enabled

<https://abcd1234.execute-api.us-west-1.amazonaws.com>**ARN**[arn:aws:apigateway:us-west-1::apis/abcd1234](#)

Stages for http-crud-tutorial-api (1)

Stage name	Invoke URL	Attached deployment	Auto deploy	Last updated
\$default	https://abcd1234.execute-api.us-west-1.amazonaws.com	abc000	enabled	2024-05-02

4. Copie a URL de invocação da sua API.

A URL completa se assemelha a `https://abcdef123.execute-api.us-west-2.amazonaws.com`.

Para criar ou atualizar um item

- Use o comando a seguir para criar ou atualizar um item. O comando inclui um corpo de solicitação com o ID, o preço e o nome do item.

```
curl -X "PUT" -H "Content-Type: application/json" -d "{\"id\": \"123\", \"price\": 12345, \"name\": \"myitem\"}" https://abcdef123.execute-api.us-west-2.amazonaws.com/items
```

Para obter todos os itens

- Use o comando a seguir para listar todos os itens.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/items
```

Para obter um item

- Use o comando a seguir para obter um item pelo seu ID.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/items/123
```

Para excluir um item

1. Use o comando a seguir para excluir um item.

```
curl -X "DELETE" https://abcdef123.execute-api.us-west-2.amazonaws.com/items/123
```

2. Obtenha todos os itens para verificar se o item foi excluído.

```
curl https://abcdef123.execute-api.us-west-2.amazonaws.com/items
```

Etapa 8: Limpar

Para evitar custos desnecessários, exclua os recursos que você criou como parte desse exercício de conceitos básicos. As etapas a seguir excluem sua API HTTP, sua função do Lambda e recursos associados.

Como excluir uma tabela do DynamoDB

1. Abra o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/>.
2. Selecionar a sua tabela.
3. Selecione Delete table (Excluir tabela).
4. Confirme a sua decisão e escolha Delete (Excluir).

Para excluir uma API HTTP

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Na página APIs , selecione uma API. Escolha Actions (Ações) e, depois, escolha Delete (Excluir).
3. Escolha Delete (Excluir).

Para excluir uma função do Lambda

1. Abra o console do Lambda em <https://console.aws.amazon.com/lambda>
2. Na página Functions(Funções), selecione uma função. Escolha Actions (Ações) e, depois, escolha Delete (Excluir).
3. Escolha Delete (Excluir).

Para excluir um grupo de logs de uma função do Lambda

1. No console do Amazon CloudWatch, abra a [página Log groups](#) (Grupos de log).
2. Na página Log groups (Grupos de log), selecione o grupo de log da função (/aws/lambda/http-crud-tutorial-function). Escolha Actions (Ações) e selecione Delete log group (Excluir grupo de log).
3. Escolha Delete (Excluir).

Para excluir a função de execução de uma função do Lambda

1. No console do AWS Identity and Access Management, abra a página [Roles](#) (Funções).
2. Selecione a atribuição da função, por exemplo, http-crud-tutorial-role.
3. Selecione Delete role (Excluir função).
4. Escolha Sim, excluir.

Próximas etapas: Automatizar com AWS SAM ou CloudFormation

Você pode automatizar a criação e a limpeza dos recursos da AWS usando o CloudFormation ou o AWS SAM. Para obter um exemplo de modelo do AWS SAM para este tutorial, consulte [template.yaml](#).

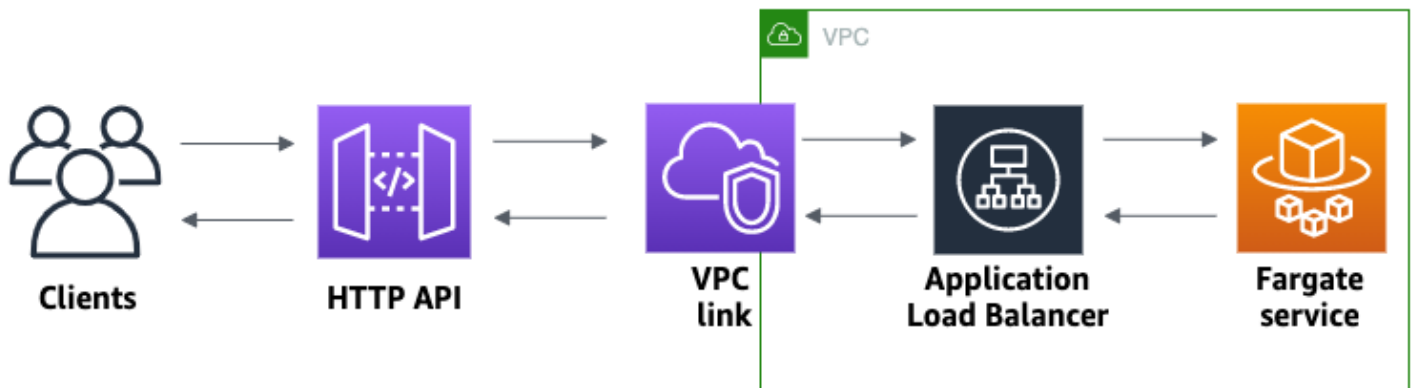
Para ver modelos de exemplo do CloudFormation, consulte [modelos de exemplo do CloudFormation](#).

Tutorial: Crie uma API HTTP com uma integração privada a um serviço do Amazon ECS

Neste tutorial, você cria uma API sem servidor que se conecta a um serviço do Amazon ECS executado em uma Amazon VPC. Clientes fora da sua Amazon VPC podem usar a API para acessar seu serviço do Amazon ECS.

Este tutorial leva aproximadamente uma hora para ser concluído. Primeiro, você usa um modelo do CloudFormation para criar um serviço da Amazon VPC e do Amazon ECS. Em seguida, você usa o console do API Gateway para criar um link da VPC. O link da VPC permite que o API Gateway acesse o serviço do Amazon ECS executado em sua Amazon VPC. Em seguida, você cria uma API HTTP que usa o link VPC para se conectar ao seu serviço do Amazon ECS. Por fim, você testa a sua API.

Quando você invoca sua API HTTP, o API Gateway encaminha a solicitação para o serviço do Amazon ECS por meio do link da VPC e, em seguida, retorna a resposta do serviço.



Para concluir esse tutorial, você precisa de uma conta da AWS e de um usuário do AWS Identity and Access Management com acesso ao console. Para obter mais informações, consulte [Pré-requisitos](#).

Neste tutorial, você usará o Console de gerenciamento da AWS. Para obter um modelo do CloudFormation que cria essa API e todos os recursos relacionados, consulte [template.yaml](#).

Tópicos

- [Etapa 1: Crie um serviço do Amazon ECS](#)
- [Etapa 2: Crie um link da VPC](#)
- [Etapa 3: Crie uma API HTTP](#)
- [Etapa 4: Crie uma rota](#)
- [Etapa 5: Crie uma integração](#)
- [Etapa 6: Teste a sua API](#)
- [Etapa 7: Limpeza](#)
- [Próximas etapas: Automatize com CloudFormation](#)

Etapa 1: Crie um serviço do Amazon ECS

O Amazon ECS é um serviço de gerenciamento de contêineres que facilita a execução, a interrupção e o gerenciamento de contêineres do Docker em um cluster. Neste tutorial, você executa seu cluster em uma infraestrutura sem servidor gerenciada pelo Amazon ECS.

Baixe e descompacte [este modelo do CloudFormation](#) que cria todas as dependências do serviço, incluindo uma Amazon VPC. Use o modelo para criar um serviço do Amazon ECS que usa um Application Load Balancer.

Como criar uma pilha do CloudFormation

1. Abra o console do CloudFormation em <https://console.aws.amazon.com/cloudformation>.
2. Selecione Create stack (Criar pilha) e With new resources (standard) (Com novos recursos (padrão)).
3. Em Specify template (Especificar modelo), escolha Upload a template file (Fazer upload de um arquivo de modelo).
4. Selecione o modelo que você baixou.
5. Escolha Next (Próximo).
6. Em Nome da pilha, insira **http-api-private-integrations-tutorial** e escolha Avançar.
7. Para Configurar opções de pilha, escolha Avançar.
8. Para Capabilities (Recursos), reconheça que CloudFormation pode criar recursos do IAM em sua conta.
9. Escolha Próximo e, depois, Enviar.

CloudFormation provisiona o serviço ECS, que pode levar alguns minutos. Quando o status da sua pilha do CloudFormation for CREATE_COMPLETE, você estará pronto para passar para a próxima etapa.

Etapa 2: Crie um link da VPC

Um link da VPC permite que o API Gateway acesse recursos privados em uma Amazon VPC. Você usa um link da VPC para permitir que os clientes acessem seu serviço do Amazon ECS por meio de sua API HTTP.

Para criar um link da VPC

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, escolha Links da VPC e selecione Criar.

Escolha o ícone do menu para se for necessário abrir o painel de navegação principal.

3. Em Escolher uma versão de link VPC, selecione Link da VPC para APIs HTTP.
4. Em Name (Nome), insira **private-integrations-tutorial**.
5. Em VPC, escolha a VPC criada na etapa 1. O nome deve começar com PrivateIntegrationsStack.
6. Para Sub-redes, selecione as duas sub-redes privadas em sua VPC. Os nomes delas terminam com PrivateSubnet.
7. Em Grupos de segurança, selecione o ID do grupo que começa com private-integrations-tutorial e tem a descrição de PrivateIntegrationsStack/PrivateIntegrationsTutorialService/Service/SecurityGroup.
8. Escolha Criar.

Depois de criar seu link VPC, o API Gateway provisiona as interfaces de rede elásticas para acessar sua VPC. O processo pode levar alguns minutos. Enquanto isso, você pode criar sua API.

Etapa 3: Crie uma API HTTP

A API HTTP fornece um endpoint HTTP para o seu serviço do Amazon ECS. Nesta etapa, você cria uma API vazia. Nas etapas 4 e 5, você configura uma rota e uma integração para conectar sua API e seu serviço do Amazon ECS.

Para criar uma API HTTP

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha Create API (Criar API) e, em seguida, em API HTTP (API HTTP), escolha Build (Criar).
3. Em API name (Nome da API), insira **http-private-integrations-tutorial**.
4. Em Tipo de endereço IP, selecione IPv4.
5. Escolha Próximo.
6. Em Configure routes (Configurar rotas), escolha Next (Próximo) para ignorar a criação da rota. Você cria rotas mais tarde.

7. Revise o estágio criado pelo API Gateway para você. O API Gateway cria um estágio `$default` com implantações automáticas ativadas, que é a melhor escolha para este tutorial. Escolha Next (Próximo).
8. Escolha Create (Criar).

Etapa 4: Crie uma rota

As rotas são uma forma de enviar as solicitações de API recebidas a recursos de backend. As rotas consistem em duas partes: um método HTTP e um caminho de recurso, por exemplo, GET `/items`. Para esta API de exemplo, criamos uma rota.

Para criar uma rota

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione a API.
3. Selecione Routes (Rotas).
4. Escolha Create (Criar).
5. Em Método, escolha **ANY**.
6. Para o caminho, insira `/{"proxy+"}`. O `{"proxy+"}` no final do caminho é uma variável de caminho ganancioso. O API Gateway envia todas as solicitações para sua API para essa rota.
7. Escolha Create (Criar).

Etapa 5: Crie uma integração

Você cria uma integração para conectar uma rota aos recursos de backend.

Para criar uma integração

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione a API.
3. Escolha Integrations (Integrações).
4. Escolha Manage integrations (Gerenciar integrações) e, em seguida, escolha Create (Criar).
5. Em Anexar esta integração a uma rota, selecione a rota ANY/`{"proxy+"}` que você criou anteriormente.
6. Em Tipo de integração, escolha Recurso privado.

7. Em Detalhes de integração, escolha Selecionar manualmente.
8. Em Serviço de destino, escolha ALB/NLB.
9. Em Load balancer (Balanceador de carga), escolha o balanceador de carga criado com o modelo do CloudFormation na Etapa 1. Seu nome deve começar com http-Priva.
10. Em Ouvinte, escolha **HTTP 80**.
11. Em Link da VPC, escolha o link da VPC que você criou na Etapa 2. O nome deve ser `private-integrations-tutorial`.
12. Escolha Create (Criar).

Para verificar se sua rota e integração estão configuradas corretamente, selecione Anexar integrações às rotas. O console mostra que você tem uma rota ANY `/{{proxy+}}` com uma integração a um Load Balancer da VPC.

Integrations

Stage: - ▼

Deploy

[Attach integrations to routes](#)[Manage integrations](#)

Routes for private-integrations-tutorial

▼ `/{{proxy+}}`

ANY

VPC Load Balancer

Integration details for route

[Detach integration](#)[Manage integration](#)ANY `/{{proxy+}}` (abc123)**Load balancer listener**

ANY HTTP:80 - http-p-Priva-ABCD1234abc

**Integration ID**

aaa111

Description

-

VPC link[efg456](#)**Timeout**

The number of milliseconds that API Gateway should wait for a response from the integration before timing out.

30000

Agora você está pronto para testar sua API.

Etapa 6: Teste a sua API

Em seguida, você testa sua API para se certificar de que ela está funcionando. Para simplificar, use um navegador da Web para invocar sua API.

Para testar sua API

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione a API.
3. Observe o URL de invocação da sua API.

my-http-api Stage: - ▼ Deploy

API details Edit

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

Stages for my-http-api (1)

🔍 Find resources

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

4. Em um navegador da Web, acesse o URL de invocação da sua API.

O URL deve ser semelhante a `https://abcdef123.execute-api.us-east-2.amazonaws.com`.

Seu navegador envia uma solicitação GET à API.

5. Verifique se a resposta da sua API é uma mensagem de boas-vindas que informa que seu aplicativo está sendo executado no Amazon ECS.

Se você vir a mensagem de boas-vindas, você criou com sucesso um serviço do Amazon ECS executado em uma Amazon VPC e usou uma API HTTP do API Gateway com um link VPC para acessar o serviço do Amazon ECS.

Etapa 7: Limpeza

Para evitar custos desnecessários, exclua os recursos que você criou como parte desse tutorial. As etapas a seguir excluem o link da VPC, a pilha do CloudFormation e a API HTTP.

Para excluir uma API HTTP

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Na página APIs , selecione uma API. Escolha Ações, escolha Excluir e, em seguida, confirme sua escolha.

Para excluir um link da VPC

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha Link da VPC.
3. Selecione seu link da VPC, escolha Excluir e confirme sua escolha.

Para excluir uma pilha do CloudFormation

1. Abra o console do CloudFormation em <https://console.aws.amazon.com/cloudformation>.
2. Selecione sua pilha do CloudFormation.
3. Escolha Excluir e, em seguida, confirme sua escolha.

Próximas etapas: Automatize com CloudFormation

Você pode automatizar a criação e a limpeza de todos os recursos da AWS envolvidos neste tutorial. Para obter um modelo do CloudFormation de exemplo completo, consulte [template.yaml](#).

Tutoriais da API WebSocket do Amazon API Gateway

Os tutoriais a seguir fornecem exercícios práticos para ajudar você a saber mais sobre as APIs de WebSocket do API Gateway.

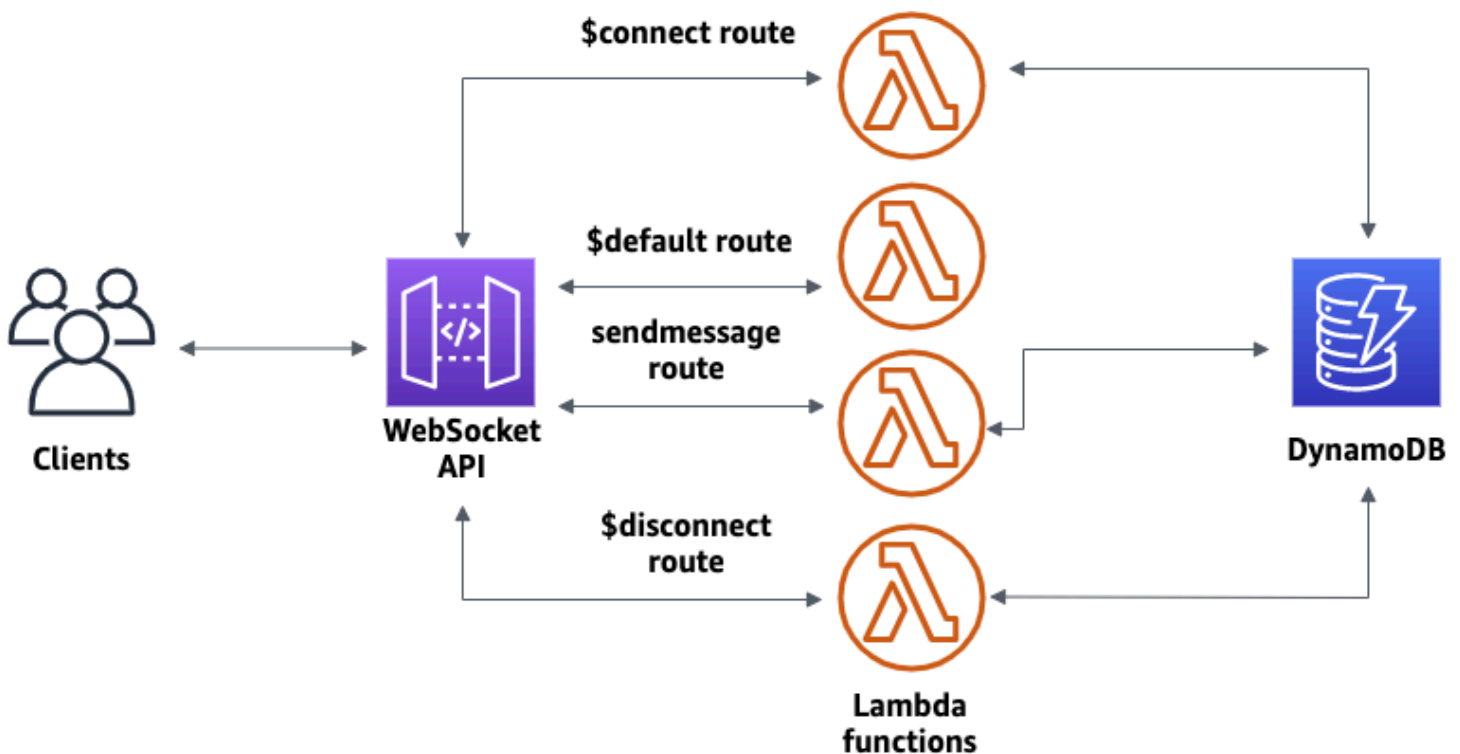
Tópicos

- [Tutorial: Crie uma aplicação de bate-papo de WebSocket com uma API de WebSocket, o Lambda e o DynamoDB](#)
- [Tutorial: Crie uma API de WebSocket com uma integração da AWS](#)

Tutorial: Crie uma aplicação de bate-papo de WebSocket com uma API de WebSocket, o Lambda e o DynamoDB

Neste tutorial, você criará uma aplicação de bate-papo sem servidor com uma API WebSocket. Com uma API WebSocket, você pode oferecer suporte à comunicação bidirecional entre clientes. Os clientes podem receber mensagens sem precisar pesquisar atualizações.

Este tutorial leva aproximadamente 30 minutos para ser concluído. Primeiro, você usará um modelo de CloudFormation para criar funções do Lambda que lidarão com solicitações de API, bem como uma tabela do DynamoDB que armazena suas IDs de cliente. Depois, você usará o console do API Gateway para criar uma API WebSocket que se integrará às suas funções do Lambda. Por fim, você testará sua API para verificar se as mensagens foram enviadas e recebidas.



Para concluir esse tutorial, você precisa de uma conta da AWS e de um usuário do AWS Identity and Access Management com acesso ao console. Para obter mais informações, consulte [Pré-requisitos](#).

Você também precisa que o `wscat` se conecte à sua API. Para obter mais informações, consulte [the section called “Use wscat para se conectar a uma API do WebSocket e enviar mensagens a ela”](#).

Tópicos

- [Etapa 1: Criar funções do Lambda e uma tabela do DynamoDB](#)
- [Etapa 2: Criar uma API WebSocket](#)

- [Etapa 3: testar sua API](#)
- [Etapa 4: limpar](#)
- [Próximas etapas: Automatize com CloudFormation](#)

Etapa 1: Criar funções do Lambda e uma tabela do DynamoDB

Baixe e descompacte [o modelo de criação de aplicativos para CloudFormation](#). Você usará esse modelo para criar uma tabela do Amazon DynamoDB para armazenar as IDs de cliente do aplicativo. Cada cliente conectado tem uma ID exclusiva que usaremos como chave de partição da tabela. Este modelo também cria funções do Lambda que atualizam suas conexões de cliente no DynamoDB e lidam com o envio de mensagens para clientes conectados.

Como criar uma pilha do CloudFormation

1. Abra o console do CloudFormation em <https://console.aws.amazon.com/cloudformation>.
2. Selecione Create stack (Criar pilha) e With new resources (standard) (Com novos recursos (padrão)).
3. Em Specify template (Especificar modelo), escolha Upload a template file (Fazer upload de um arquivo de modelo).
4. Selecione o modelo que você baixou.
5. Escolha Next (Próximo).
6. Em Nome da pilha, insira **websocket-api-chat-app-tutorial** e escolha Avançar.
7. Para Configurar opções de pilha, escolha Avançar.
8. Para Capabilities (Recursos), reconheça que CloudFormation pode criar recursos do IAM em sua conta.
9. Escolha Próximo e, depois, Enviar.

CloudFormationO fornece os recursos especificados no modelo. Pode demorar alguns minutos para concluir o provisionamento de seus recursos. Quando o status da sua pilha do CloudFormation for CREATE_COMPLETE, você estará pronto para passar para a próxima etapa.

Etapa 2: Criar uma API WebSocket

Você criará uma API WebSocket para lidar com conexões de clientes e rotear solicitações para as funções do Lambda que criou na Etapa 1.

Como criar uma API WebSocket

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione Create API (Criar API). Depois, em WebSocket API (API WebSocket), escolha Build (Criar).
3. Em API name (Nome da API), insira **websocket-chat-app-tutorial**.
4. Em Tipo de endereço IP, selecione IPv4.
5. Em Route selection expression (Expressão de seleção de rota), insira **request.body.action**. A expressão de seleção de rota determina qual rota o API Gateway invoca quando um cliente envia uma mensagem.
6. Escolha Próximo.
7. Em Predefined routes (Rotas predefinidas), escolha Add \$connect (Adicionar \$connect), Add \$disconnect (Adicionar \$disconnect) e Add \$default (Adicionar \$default). As rotas \$connect e \$disconnect são rotas especiais que o API Gateway invoca automaticamente quando um cliente se conecta ou se desconecta de uma API. O API Gateway invoca a rota \$default quando nenhuma outra rota corresponde a uma solicitação.
8. Em Custom routes (Rotas personalizadas), escolha Add custom route (Adicionar rota personalizada). Em Route key (Chave de rota), insira **sendMessage**. Essa rota personalizada lida com mensagens enviadas para clientes conectados.
9. Escolha Avançar.
10. Em Attach integrations (Anexar integrações), para cada rota e Integration type (Tipo de integração), escolha Lambda.

Em Lambda, escolha a função do Lambda correspondente que você criou com o CloudFormation na Etapa 1. O nome de cada função corresponde a uma rota. Por exemplo, para a rota \$connect, escolha a função chamada **websocket-chat-app-tutorial-ConnectHandler**.

11. Revise o estágio criado pelo API Gateway para você. Por padrão, o API Gateway cria um nome de estágio **production** e implanta automaticamente sua API nesse estágio. Escolha Próximo.
12. Selecione Create and deploy (Criar e implantar).

Etapa 3: testar sua API

Depois, você testará sua API para verificar se ela está funcionando corretamente. Use o comando **wscat** para se conectar à API.

Como obter a URL para invocar a sua API

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione a API.
3. Escolha Stages (Estágios) e, depois, production (produção).
4. Observe o URL WebSocket da API. O URL deve ser semelhante a `wss://abcdef123.execute-api.us-east-2.amazonaws.com/production`.

Como se conectar à sua API

1. Use o comando a seguir para se conectar à sua API. Quando você se conecta à API, o API Gateway invoca a rota `$connect`. Quando essa rota é invocada, ela chama uma função do Lambda que armazena seu ID de conexão no DynamoDB.

```
wscat -c wss://abcdef123.execute-api.us-west-2.amazonaws.com/production
```

```
Connected (press CTRL+C to quit)
```

2. Abra um novo terminal e execute o comando `wscat` novamente com os parâmetros a seguir.

```
wscat -c wss://abcdef123.execute-api.us-west-2.amazonaws.com/production
```

```
Connected (press CTRL+C to quit)
```

Isso fornece dois clientes conectados que podem trocar mensagens.

Para enviar uma mensagem

- O API Gateway determina qual rota invocar com base na expressão de seleção de rotas da API. A expressão de seleção de rotas da sua API é `$request.body.action`. Como resultado, o API Gateway invoca a rota `sendmessage` quando você envia a seguinte mensagem:

```
{"action": "sendmessage", "message": "hello, everyone!"}
```

A função do Lambda associada à rota invocada coleta os IDs do cliente do DynamoDB. Depois, a função chama a API de gerenciamento do API Gateway e envia a mensagem para esses clientes. Todos os clientes conectados recebem a seguinte mensagem:

```
< hello, everyone!
```

Como invocar a rota \$default da API

- O API Gateway invocará a rota padrão da API quando um cliente envia uma mensagem que não corresponde às rotas definidas. A função do Lambda associada à rota \$default usa a API de gerenciamento do API Gateway para enviar informações do cliente sobre sua conexão.

```
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","connect
```

Para se desconectar de sua API

- Pressione **CTRL+C** para se desconectar de sua API. Quando um cliente se desconecta da sua API, o API Gateway invoca a rota \$disconnect de sua API. A integração do Lambda para a rota \$disconnect de sua API remove o ID da conexão do DynamoDB.

Etapa 4: limpar

Para evitar custos desnecessários, exclua os recursos que você criou como parte desse tutorial. As etapas a seguir excluem a API WebSocket e sua pilha do CloudFormation.

Como excluir uma API WebSocket

- Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
- Na página APIs, selecione sua API do websocket-chat-app-tutorial. Escolha Ações, escolha Excluir e, em seguida, confirme sua escolha.

Para excluir uma pilha do CloudFormation

1. Abra o console do CloudFormation em <https://console.aws.amazon.com/cloudformation>.
2. Selecione sua pilha do CloudFormation.
3. Escolha Excluir e, em seguida, confirme sua escolha.

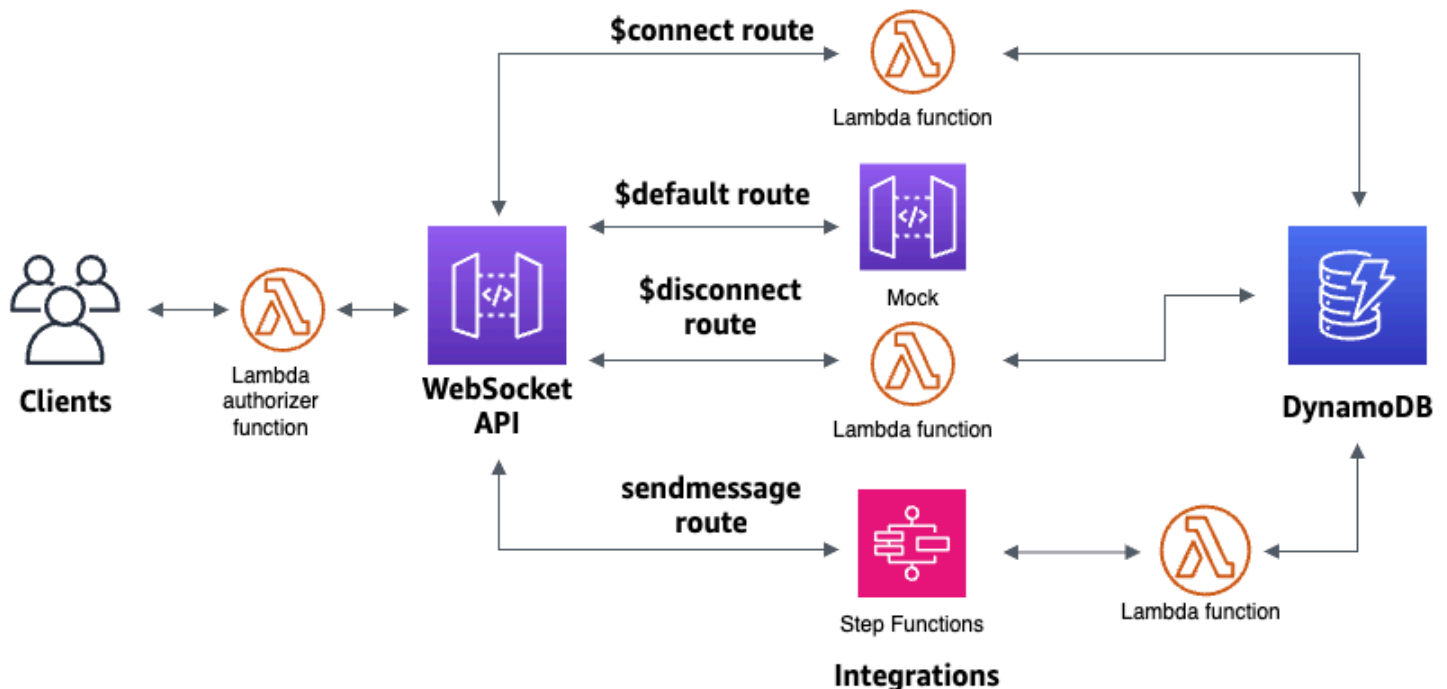
Próximas etapas: Automatize com CloudFormation

Você pode automatizar a criação e a limpeza de todos os recursos da AWS envolvidos neste tutorial. Para ver um modelo do CloudFormation que crie essa API e todos os recursos relacionados, consulte [ws-chat-app.yaml](https://github.com/aws-samples/ws-chat-app).

Tutorial: Crie uma API de WebSocket com uma integração da AWS

Neste tutorial, você criará uma aplicação de transmissão com tecnologia sem servidor com uma API de WebSocket. Os clientes podem receber mensagens sem precisar pesquisar atualizações.

Este tutorial mostra como transmitir mensagens para clientes conectados e inclui um exemplo de um autorizador do Lambda, uma integração simulada e uma integração sem proxy com o Step Functions.



Depois de criar seus recursos usando um modelo do CloudFormation, você usará o console do API Gateway para criar uma API de WebSocket que se integrará aos seus recursos da AWS. Você

anexará um autorizador do Lambda à sua API e criará uma integração de serviços da AWS com o Step Functions para iniciar a execução de uma máquina de estado. A máquina de estado do Step Functions invocará uma função do Lambda que envia uma mensagem para todos os clientes conectados.

Depois de criar sua API, você testará sua conexão com a API e verificará se as mensagens foram enviadas e recebidas. Este tutorial leva aproximadamente 45 minutos para ser concluído.

Tópicos

- [Pré-requisitos](#)
- [Etapa 1: Criar recursos](#)
- [Etapa 2: Criar uma API WebSocket](#)
- [Etapa 3: Criar um autorizador do Lambda](#)
- [Etapa 4: Criar uma integração bidirecional simulada](#)
- [Etapa 5: Criar uma integração sem proxy com o Step Functions](#)
- [Etapa 6: Teste a sua API](#)
- [Etapa 7: limpar](#)
- [Próximas etapas](#)

Pré-requisitos

Você precisa dos seguintes pré-requisitos:

- Uma conta da AWS e um usuário do AWS Identity and Access Management com acesso ao console. Para obter mais informações, consulte [Pré-requisitos](#).
- `wscat` para se conectar à sua API. Para obter mais informações, consulte [the section called “Use wscat para se conectar a uma API do WebSocket e enviar mensagens a ela”](#).

Recomendamos que você conclua o tutorial de aplicação de bate-papo do WebSocket antes de iniciar este tutorial. Para concluir o tutorial de aplicação de bate-papo do WebSocket, consulte [the section called “Tutorial: Crie uma aplicação de bate-papo de WebSocket”](#).

Etapa 1: Criar recursos

Baixe e descompacte [o modelo de criação de aplicativos para CloudFormation](#). Você usará este modelo para criar o seguinte:

- Funções do Lambda que lidam com solicitações de API e autorizam o acesso à sua API.
- Uma tabela do DynamoDB para armazenar IDs de clientes e a identificação do usuário da entidade principal retornada pelo autorizador do Lambda.
- Uma máquina de estado do Step Functions para enviar mensagens para clientes conectados.

Como criar uma pilha do CloudFormation

1. Abra o console do CloudFormation em <https://console.aws.amazon.com/cloudformation>.
2. Selecione Create stack (Criar pilha) e With new resources (standard) (Com novos recursos (padrão)).
3. Em Specify template (Especificar modelo), escolha Upload a template file (Fazer upload de um arquivo de modelo).
4. Selecione o modelo que você baixou.
5. Escolha Next (Próximo).
6. Em Nome da pilha, insira **websocket-step-functions-tutorial** e escolha Avançar.
7. Para Configurar opções de pilha, escolha Avançar.
8. Para Capabilities (Recursos), reconheça que CloudFormation pode criar recursos do IAM em sua conta.
9. Escolha Próximo e, depois, Enviar.

CloudFormationO fornece os recursos especificados no modelo. Pode demorar alguns minutos para concluir o provisionamento de seus recursos. Escolha a guia Saídas para ver os recursos criados e seus ARNs. Quando o status da sua pilha do CloudFormation for CREATE_COMPLETE, você estará pronto para passar para a próxima etapa.

Etapa 2: Criar uma API WebSocket

Você criará uma API de WebSocket para lidar com conexões de clientes e rotear solicitações para os recursos criados na Etapa 1.

Como criar uma API WebSocket

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione Criar API. Depois, em WebSocket API (API WebSocket), escolha Build (Criar).

3. Em API name (Nome da API), insira **websocket-step-functions-tutorial**.
4. Em Tipo de endereço IP, selecione IPv4.
5. Em Route selection expression (Expressão de seleção de rota), insira **request.body.action**.

A expressão de seleção de rota determina qual rota o API Gateway invoca quando um cliente envia uma mensagem.

6. Escolha Next (Próximo).
7. Em Rotas predefinidas, escolha Adicionar \$connect, Adicionar \$disconnect e Adicionar \$default.

As rotas \$connect e \$disconnect são rotas especiais que o API Gateway invoca automaticamente quando um cliente se conecta ou se desconecta de uma API. O API Gateway invoca a rota \$default quando nenhuma outra rota corresponde a uma solicitação. Você criará uma rota personalizada para se conectar ao Step Functions depois de criar sua API.

8. Escolha Next (Próximo).
9. Para Integração para \$connect, faça o seguinte:
 - a. Em Tipo de integração, escolha Lambda.
 - b. Em Função do Lambda, escolha a função do Lambda \$connect correspondente que você criou com o CloudFormation na Etapa 1. O nome da função do Lambda deve começar com **websocket-step**.
10. Para Integração para \$disconnect, faça o seguinte:
 - a. Em Tipo de integração, escolha Lambda.
 - b. Em Função do Lambda, escolha a função do Lambda \$disconnect correspondente que você criou com o CloudFormation na Etapa 1. O nome da função do Lambda deve começar com **websocket-step**.
11. Em Integração para \$default, escolha mock.

Em uma integração simulada, o API Gateway gerencia a resposta da rota sem um back-end de integração.

12. Escolha Next (Próximo).
13. Revise o estágio criado pelo API Gateway para você. Por padrão, o API Gateway cria um estágio nomeado production e implanta automaticamente sua API nesse estágio. Escolha Next (Próximo).
14. Selecione Create and deploy (Criar e implantar).

Etapa 3: Criar um autorizador do Lambda

Para controlar o acesso à sua API de WebSocket, você cria um autorizador do Lambda. O modelo do CloudFormation criou a função de autorizador do Lambda para você. Você pode criar a função do Lambda no console do Lambda. O nome deve começar com **websocket-step-functions-tutorial-AuthORIZERHandler**. Essa função do Lambda nega todas as chamadas para a API de WebSocket, a menos que o cabeçalho Authorization seja Allow. A função do Lambda também transmite a variável `$context.authorizer.principalId` para sua API, que é usada posteriormente na tabela do DynamoDB para identificar os chamadores da API.

Nesta etapa, você configura a rota `$connect` para usar o autorizador do Lambda.

Para criar um autorizador do Lambda

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, selecione Autorizadores.
3. Selecione Criar um autorizador.
4. Em Nome do autorizador, insira **LambdaAuthORIZER**.
5. Em ARN do autorizador, insira o nome do autorizador criado pelo modelo CloudFormation. O nome deve começar com **websocket-step-functions-tutorial-AuthORIZERHandler**.

Note

Recomendamos que você não use este exemplo de autorizador para suas APIs de produção.

6. Em Identificar tipo de origem, escolha Cabeçalho. Em Chave, digite **Authorization**.
7. Selecione Criar autorizador.

Depois de criar o autorizador, você deve anexá-lo à rota `$connect` da sua API.

Para anexar um autorizador à rota `$connect`

1. No painel de navegação principal, selecione Rotas.
2. Escolha a rota `$connect`.
3. Na seção Configurações de solicitação de rota, selecione Editar.
4. Em Autorização, escolha o menu suspenso e selecione o autorizador da solicitação.

5. Escolha Salvar alterações.

Etapa 4: Criar uma integração bidirecional simulada

Em seguida, crie a integração simulada bidirecional para a rota `$default`. Uma integração simulada permite que você envie uma resposta ao cliente sem usar um back-end. Ao criar uma integração para a rota `$default`, você pode mostrar aos clientes como interagir com sua API.

Você configura a rota `$default` a fim de informar os clientes para que usem a rota `sendMessage`.

Para criar uma integração simulada

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha a rota `$default` e a guia Solicitação de integração.
3. Em Modelos de solicitação, escolha Editar.
4. Em Expressão de seleção de modelo, insira **200** e escolha Editar.
5. Na guia Solicitação de integração, em Modelos de solicitação, escolha Criar modelo.
6. Em Chave do modelo, insira **200**.
7. Em Gerar modelo, insira o seguinte modelo de mapeamento:

```
{"statusCode": 200}
```

Selecione Criar modelo.

O resultado deve ser algo semelhante a:

Route request
Integration request
Integration response
Route response

Integration request settings

Integration type [Info](#)
Mock

Timeout
29000 ms

[Edit](#)

Request templates (1)

[Edit](#)
[Create template](#)

Use request templates to transform the incoming message before sending it to the integration. API Gateway uses a template selection expression to determine which template to use. Name the template with a key that matches the result of the selection expression.

Template selection expression
200

200

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

```

1  {"statusCode": 200}
2
3

```

8. No painel rota \$default, escolha Habilitar comunicação bidirecional.
9. Escolha a guia Resposta de integração e Criar resposta de integração.
10. Em Chave de resposta, insira **\$default**.
11. Em Expressão de seleção de modelos, insira **200**.
12. Selecione Criar resposta.
13. Em Modelos de resposta, selecione Criar modelo.
14. Em Chave do modelo, insira **200**.
15. Em Modelo de resposta, insira o seguinte modelo de mapeamento:

```

{"Use the sendmessage route to send a message. Connection ID:
 $context.connectionId"}

```

16. Selecione Criar modelo.

O resultado deve ser algo semelhante a:

Route request

Integration request

Integration response

Route response

Integration response settings

Create integration response

Integration responses allow you to configure transformations on the outgoing message's payload using response template definitions. The response chosen is based on the response key found in the outgoing message after evaluating the response selection expression.

\$default

EditDelete

Template selection expression
200

Response templates

Create template

200

EditDelete

```
1  {"Use the sendmessage route to send a message. Connection ID:
2  $context.connectionId"}
3
```

Etapa 5: Criar uma integração sem proxy com o Step Functions

Em seguida, crie uma rota sendmessage. Os clientes podem invocar a rota sendmessage para transmitir uma mensagem para todos os clientes conectados. A rota sendmessage tem uma integração de serviço da AWS sem proxy com o AWS Step Functions. A integração invoca o comando [StartExecution](#) para a máquina de estado do Step Functions que o modelo do CloudFormation criou para você.

Para criar uma integração sem proxy

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.

2. Escolha Create route (Criar rota).
3. Em Route key (Chave de rota), insira **sendmessage**.
4. Em Tipo de integração, selecione Serviço da AWS .
5. Em Região da AWS, insira a região em que você implantou seu modelo do CloudFormation.
6. Em Serviço da AWS, escolha Step Functions.
7. Em Método HTTP, escolha POST.
8. Em Nome da ação, insira **StartExecution**.
9. Em Perfil de execução, insira o perfil de execução criado pelo modelo do CloudFormation. O nome deve ser WebSocketTutorialApiRole.
10. Escolha Create route (Criar rota).

Em seguida, crie um modelo de mapeamento para enviar parâmetros de solicitação para a máquina de estado do Step Functions.

Como criar um modelo de mapeamento

1. Escolha a rota sendmessage e a guia Solicitação de integração.
2. Na seção Modelos de solicitação, selecione Editar.
3. Em Expressão de seleção de modelos, insira **\\$default**.
4. Escolha Editar.
5. Na seção Modelos de solicitação, selecione Criar modelo.
6. Em Chave do modelo, insira **\\$default**.
7. Em Gerar modelo, insira o seguinte modelo de mapeamento:

```
#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"
}
```

Substitua *stateMachineArn* pelo ARN da máquina de estado criado pelo CloudFormation.

O modelo de mapeamento faz o seguinte:

- Cria a variável `$domain` usando a variável de contexto `domainName`.
- Cria a variável `$stage` usando a variável de contexto `stage`.

As variáveis `$domain` e `$stage` são necessárias para criar um URL de retorno de chamada.

- Recebe a mensagem JSON `sendmessage` recebida e extrai a propriedade `message`.
- Cria a entrada para a máquina de estado. A entrada é o domínio e o estágio da API de WebSocket e a mensagem da rota `sendmessage`.

8. Selecione Criar modelo.

Request templates (1)

[Edit](#)[Create template](#)

Use request templates to transform the incoming message before sending it to the integration. API Gateway uses a template selection expression to determine which template to use. Name the template with a key that matches the result of the selection expression.

Template selection expression

`\$default`

`\$default`

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

```
1  #set($domain = "$context.domainName")
2  #set($stage = "$context.stage")
3  #set($body = $input.json('$'))
4  #set($getMessage = $util.parseJson($body))
5  #set($mymessage = $getMessage.message)
6  {
7    "input": "{\"domain\": \"$domain\", \"stage\": \"$stage\", \"message\": \"$mymessage\"}",
8    "stateMachineArn": "arn:aws:states:us-east-2:111222333:stateMachine:WebSocket-Tutorial-StateMachine"
9  }
```

Você pode criar uma integração sem proxy nas rotas \$connect ou \$disconnect para adicionar ou remover diretamente um ID de conexão da tabela do DynamoDB, sem invocar uma função do Lambda.

Etapa 6: Teste a sua API

Em seguida, implante e teste sua API para verificar se ela está funcionando corretamente. Você usará o comando `wscat` para se conectar à API e usará um comando de barra para enviar uma estrutura ping para verificar a conexão com a API de WebSocket.

Para implantar sua API

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, selecione Rotas.
3. Escolha Implantar API.
4. Em Estágio, escolha production.
5. (Opcional) Em Descrição da implantação, insira uma descrição.
6. Escolha Implantar.

Depois de implantar sua API, você pode invocá-la. Use o URL de invocação para chamar sua API.

Para obter o URL de invocação da sua API

1. Selecione a API.
2. Escolha Stages (Estágios) e, depois, production (produção).
3. Observe o URL WebSocket da API. O URL deve ser semelhante a `wss://abcdef123.execute-api.us-east-2.amazonaws.com/production`.

Agora que você tem o URL de invocação, já pode testar a conexão com a API de WebSocket.

Para testar a conexão com sua API

1. Use o comando a seguir para se conectar à sua API. Primeiro, teste a conexão invocando o caminho `/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. Insira o comando a seguir para fazer ping do frame do controle. Você pode usar uma estrutura de controle para fins de manutenção de atividade do lado do cliente.

```
/ping
```

O resultado deve ser algo semelhante a:

```
< Received pong (data: "")
```

Agora que já testou a conexão, pode testar se a sua API funciona corretamente. Nesta etapa, abra uma nova janela de terminal para que a API de WebSocket possa enviar uma mensagem para todos os clientes conectados.

Para testar sua API

1. Abra um novo terminal e execute o comando `wscat` novamente com os parâmetros a seguir.

```
wscat -c wss://abcdef123.execute-api.us-east-2.amazonaws.com/production -H  
"Authorization: Allow"
```

```
Connected (press CTRL+C to quit)
```

2. O API Gateway determina qual rota invocar com base na expressão de seleção da solicitação de rotas da API. A expressão de seleção de rotas da sua API é `$request.body.action`. Como resultado, o API Gateway invoca a rota `sendmessage` quando você envia a seguinte mensagem:

```
{"action": "sendmessage", "message": "hello, from Step Functions!"}
```

A máquina de estado do Step Functions associada à rota invoca uma função do Lambda com a mensagem e o URL de retorno de chamada. Depois, a função do Lambda chama a API de gerenciamento do API Gateway e envia a mensagem para todos os clientes conectados. Todos os clientes conectados recebem a seguinte mensagem:

```
< hello, from Step Functions!
```

Agora que você testou sua API de WebSocket, pode se desconectar da sua API.

Para se desconectar de sua API

- Pressione CTRL+C para se desconectar de sua API.

Quando um cliente se desconecta da sua API, o API Gateway invoca a rota \$disconnect da sua API. A integração do Lambda para a rota \$disconnect da sua API remove o ID da conexão do DynamoDB.

Etapa 7: limpar

Para evitar custos desnecessários, exclua os recursos que você criou como parte desse tutorial. As etapas a seguir excluem a API WebSocket e sua pilha do CloudFormation.

Como excluir uma API WebSocket

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Na página APIs, selecione websocket-api.
3. Escolha Ações, escolha Excluir e, em seguida, confirme sua escolha.

Para excluir uma pilha do CloudFormation

1. Abra o console do CloudFormation em <https://console.aws.amazon.com/cloudformation>.
2. Selecione sua pilha do CloudFormation.
3. Escolha Excluir e, em seguida, confirme sua escolha.

Próximas etapas

Você pode automatizar a criação e a limpeza de todos os recursos da AWS envolvidos neste tutorial. Para ver um exemplo de um modelo do CloudFormation que automatiza essas ações neste tutorial, consulte [ws-sfn.zip](#).

APIs REST do API Gateway

Uma API REST no API Gateway é uma coleção de recursos e métodos que são integrados aos endpoints HTTP de back-end, funções do Lambda ou outros serviços da AWS. É possível usar os recursos do API Gateway para obter ajuda em todos os aspectos do ciclo de vida da API, desde a criação até o monitoramento das APIs de produção.

As APIs REST do API Gateway usam um modelo de solicitação/resposta em que um cliente envia uma solicitação para um serviço, e o serviço responde de forma síncrona. Esse tipo de modelo é adequado para muitos tipos diferentes de aplicativos que dependem de comunicação síncrona.

Tópicos

- [Desenvolver APIs REST no API Gateway](#)
- [Publicar APIs REST para os clientes invocarem](#)
- [Otimizar o desempenho das APIs REST](#)
- [Distribuir as APIs REST para clientes no API Gateway](#)
- [Proteger as APIs REST no API Gateway](#)
- [Monitorar APIs REST no API Gateway](#)

Desenvolver APIs REST no API Gateway

No Amazon API Gateway, você cria uma API REST como uma coleção de entidades programáveis conhecida como [recursos](#) do API Gateway. Por exemplo, você usa um recurso [RestApi](#) para representar uma API que pode conter uma coleção de entidades [Recurso](#).

Cada entidade Resource pode ter um ou mais recursos [Method](#). Um Method é uma solicitação recebida enviada pelo cliente e pode conter os seguintes parâmetros de solicitação: um parâmetro de caminho, um cabeçalho ou um parâmetro de string de consulta. Além disso, dependendo do método HTTP, a solicitação pode conter um corpo. O método define como o cliente acessa o Resource exposto. Para integrar o Method a um endpoint de back-end, também conhecido como endpoint de integração, crie um recurso [Integration](#). Isso encaminha a solicitação recebida para um URI de endpoint de integração especificado. Se necessário, é possível transformar os parâmetros ou o corpo da solicitação para atender aos requisitos de backend, ou criar uma integração de proxy, na qual o API Gateway envia a solicitação inteira em um formato padronizado ao URI do endpoint de integração e, depois, envia diretamente a resposta ao cliente.

Para as respostas, é possível criar um recurso [MethodResponse](#) para representar uma resposta recebida pelo cliente e criar um recurso [IntegrationResponse](#) para representar a resposta retornada pelo backend. Use uma resposta de integração para transformar os dados da resposta do backend antes de retornar os dados ao cliente ou transmitir a resposta do backend da forma em que se encontra para o cliente.

Exemplo de recurso para uma API REST

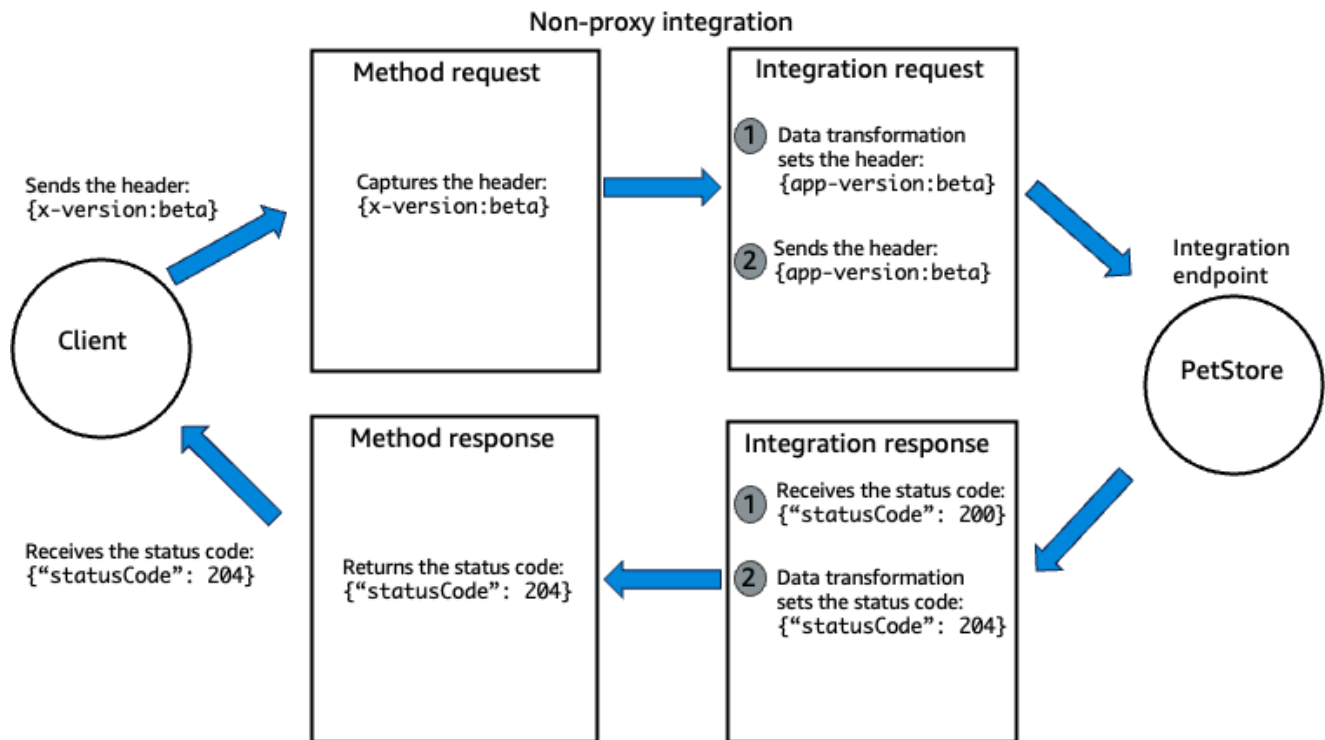
O diagrama a seguir mostra como o API Gateway implementa esse modelo de solicitação/resposta para uma integração de proxy HTTP e uma integração HTTP sem proxy para o recurso GET `/pets`. O cliente envia o cabeçalho `x-version:beta` ao API Gateway e o API Gateway envia o código de status 204 ao cliente.

Na integração sem proxy, o API Gateway realiza transformações de dados para atender aos requisitos de backend, modificando a solicitação e a resposta da integração. Em uma integração sem proxy, é possível acessar o corpo na solicitação do método, mas ele é transformado na solicitação de integração. Quando o endpoint de integração retorna uma resposta com um corpo, você a acessa e a transforma na resposta de integração. Não é possível modificar o corpo na resposta do método.

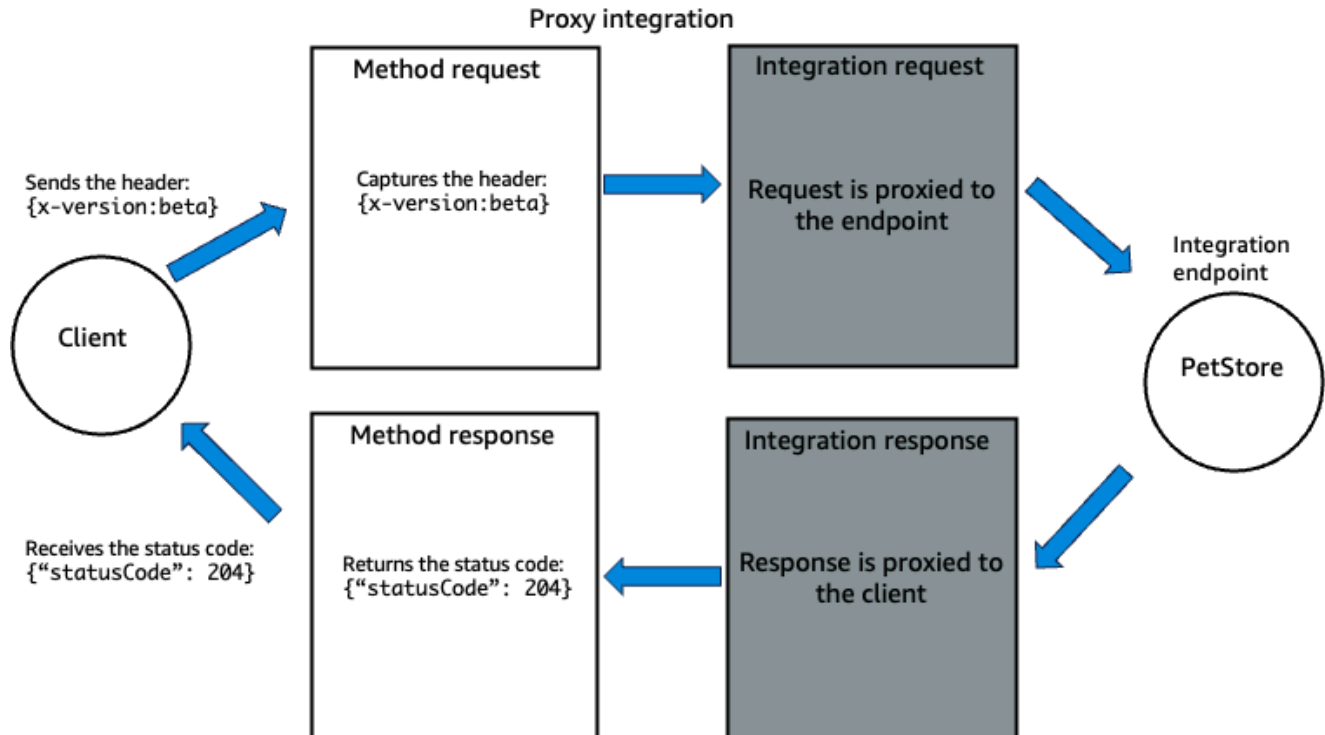
Na integração de proxy, o endpoint de integração modifica a solicitação e a resposta. O API Gateway não modifica a solicitação nem a resposta de integração e envia a solicitação recebida ao backend no estado em que se encontra.

Independentemente do tipo de integração, o cliente enviou uma solicitação ao API Gateway, que a respondeu de forma síncrona.

Non-proxy integration



Proxy integration



Os exemplos de logs de execução a seguir mostram o que o API Gateway registraria no exemplo anterior. Para garantir maior clareza, alguns valores e logs iniciais foram removidos:

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
```

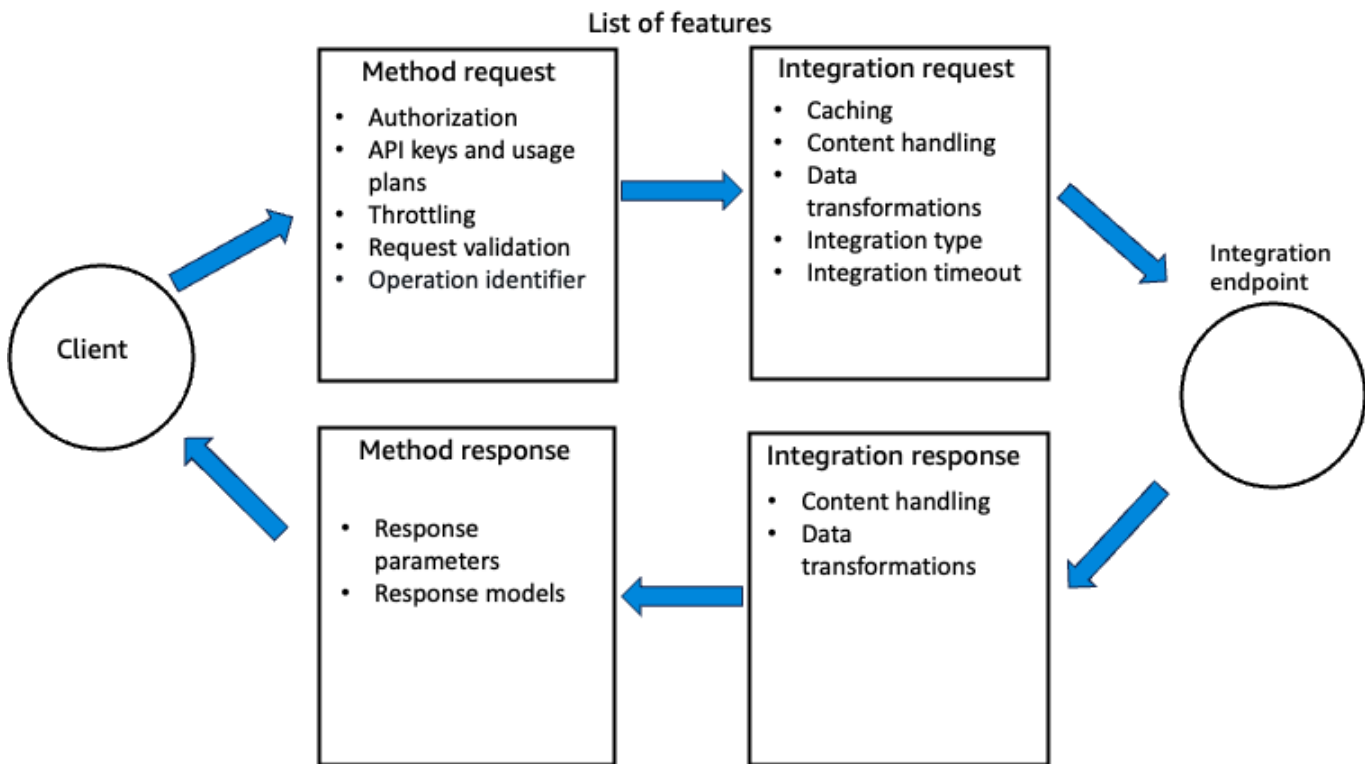
Para importar uma API semelhante e testá-la no Console de gerenciamento da AWS, consulte o [exemplo de API](#).

Recursos adicionais da API REST para desenvolvimento

O API Gateway é compatível com recursos adicionais para o desenvolvimento da API REST. Por exemplo, para ajudar os clientes a compreenderem sua API, você pode fornecer a documentação da API. Para fazer isso, adicione um recurso [DocumentationPart](#) para uma entidade de API compatível.

Para controlar como os clientes chamam uma API, use [permissões do IAM](#), um [autorizador de Lambda](#) ou um [grupo de usuários Amazon Cognito](#). Para medir o uso da sua API, configure [planos de uso](#) para limitar as solicitações de API. Você pode habilitá-los ao criar ou atualizar a API.

O diagrama a seguir mostra os recursos disponíveis para o desenvolvimento da API REST e onde, no modelo de solicitação/resposta, esses recursos são configurados.



Para conferir uma introdução sobre como criar uma API, consulte [the section called “Tutorial: Crie uma API REST com uma integração de proxy do Lambda”](#). Para obter mais informações sobre os recursos do API Gateway que você pode usar ao desenvolver uma API REST, consulte os tópicos a seguir. Esses tópicos contêm informações conceituais e procedimentos que você pode executar usando o console do API Gateway, a API REST do API Gateway, a AWS CLI ou um dos AWS SDKs.

Tópicos

- [Tipos de endpoint da API para APIs REST no API Gateway](#)
- [Políticas de segurança para APIs REST no API Gateway](#)
- [Tipos de endereço IP para APIs REST no API Gateway](#)
- [Métodos para APIs REST no API Gateway](#)
- [Controlar e gerenciar o acesso a APIs REST no API Gateway](#)
- [Integrações para APIs REST no API Gateway](#)
- [Solicitar validação para APIs REST no API Gateway](#)
- [Transformações de dados para APIs REST no API Gateway](#)
- [Respostas do gateway para APIs REST no API Gateway](#)
- [CORS para APIs REST no API Gateway](#)

- [Tipos de mídia binários para APIs REST no API Gateway](#)
- [Invocar APIs REST no API Gateway](#)
- [Desenvolver APIs REST usando OpenAPI no API Gateway](#)

Tipos de endpoint da API para APIs REST no API Gateway

Um tipo de [endpoint da API](#) refere-se ao nome do host da API. O tipo de endpoint da API pode ser edge-optimized, regional ou privado, dependendo de onde a maior parte do seu tráfego de API se origina.

Endpoint de API otimizado para bordas

Um [endpoint de API otimizado para borda](#) geralmente direciona as solicitações para o ponto de presença (POP) do CloudFront mais próximo, o que pode ajudar nos casos em que os clientes estão distribuídos geograficamente. Esse é o tipo de endpoint padrão para APIs REST do API Gateway.

As APIs otimizadas para fronteiras mantêm em letra maiúscula os nomes dos [cabeçalhos HTTP](#) (por exemplo, Cookie).

O CloudFront classifica os cookies HTTP em ordem natural por nome de cookie antes de encaminhar a solicitação para sua origem. Para obter mais informações sobre a maneira como o CloudFront processa os cookies, consulte [Armazenamento em cache de conteúdo com base em cookies](#).

Qualquer nome de domínio personalizado que for usado para uma API otimizada para a borda se aplicará a todas as regiões.

Endpoints de API regionais

Um [endpoint de API regional](#) é destinado a clientes na mesma região. Quando um cliente em execução em uma instância do EC2 chama uma API na mesma região ou quando uma API é destinada a atender a um pequeno número de clientes com alta demanda, uma API regional reduz a sobrecarga da conexão.

Para uma API regional, o nome de domínio personalizado que você usa é específico da região em que a API é implantada. Se você implantar uma API regional implantada em várias regiões, o nome de domínio personalizado poderá ser o mesmo em todas as regiões. Você pode usar domínios personalizados em conjunto com o Amazon Route 53 para executar tarefas como [roteamento baseado em latência](#). Para obter mais informações, consulte [the section called “Configurar um](#)

[nome de domínio regional personalizado](#) e [the section called “Configurar um nome de domínio personalizado otimizado para borda no API Gateway”](#).

Os endpoints de API regionais transmitem todos os nomes de cabeçalho no estado em que se encontram.

 Note

Nos casos em que os clientes da API são geograficamente dispersos, ainda poderá fazer sentido usar um endpoint de API regional junto à sua própria distribuição do Amazon CloudFront para garantir que o API Gateway não associe a API às distribuições do CloudFront controladas pelo serviço. Para obter mais informações sobre esse caso de uso, consulte [Como posso configurar o API Gateway com minha própria distribuição do CloudFront?](#).

Endpoints privados de API

Um [endpoint privado da API](#) é um endpoint de API que somente pode ser acessado de sua Amazon Virtual Private Cloud (VPC) usando um VPC endpoint de interface, uma endpoint network interface (ENI – Interface de rede de endpoint) que você cria em sua VPC. Para obter mais informações, consulte [the section called “APIs REST privadas”](#).

Os endpoints privados de API transmitem todos os nomes de cabeçalho no estado em que se encontram.

Alterar um tipo de endpoint de API pública ou privada no API Gateway

Alterar o tipo de endpoint da API requer que você atualize a configuração da API. É possível alterar um tipo de API existente usando o console do API Gateway, a AWS CLI ou um SDK da AWS para API Gateway. O tipo de endpoint não poderá ser alterado novamente até que a alteração atual seja concluída, mas a API estará disponível.

Há suporte para as seguintes alterações nos tipos de endpoints:

- De “otimizada para borda” para “regional” ou “privada”
- De “regional” para “otimizada para borda” ou “privada”
- De “privada” para “regional”

Não é possível alterar uma API privada para uma API otimizada para fronteiras.

Se você está alterando uma API pública do tipo “otimizada para borda” para “regional” ou vice-versa, observe que uma API otimizada para borda pode ter comportamentos diferentes em comparação com uma API regional. Por exemplo, uma API otimizada para fronteiras remove o cabeçalho Content-MD5. Qualquer valor de hash MD5 transmitido para o backend pode ser expresso em um parâmetro de string de solicitação ou uma propriedade de corpo. No entanto, a API regional transmite esse cabeçalho, embora ela possa remapear o nome do cabeçalho para outro nome. A compreensão das diferenças ajuda você a decidir como atualizar de uma API otimizada para borda para uma regional ou de uma API regional para uma otimizada para borda.

Tópicos

- [Usar o console do API Gateway para alterar um tipo de endpoint de API](#)
- [Usar a AWS CLI para alterar um tipo de endpoint de API](#)

Usar o console do API Gateway para alterar um tipo de endpoint de API

Para alterar o tipo de endpoint de API da sua API, realize um dos seguintes conjuntos de etapas:

Como converter um endpoint público de regional em otimizado para borda e vice-versa

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST.
3. Escolha Configurações da API.
4. Na seção Detalhes da API, escolha Editar.
5. Em Tipo de endpoint da API, selecione Otimizado para borda ou Regional.
6. Escolha Salvar alterações.
7. Reimplante sua API para que as alterações sejam aplicadas.

Converter um endpoint privado em um endpoint regional

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST.
3. Edite a política de recursos da sua API para remover qualquer menção de VPCs ou endpoints da VPC, a fim de que as chamadas de API de fora ou de dentro da sua VPC sejam bem-sucedidas.

4. Escolha Configurações da API.
5. Na seção Detalhes da API, escolha Editar.
6. Em Tipo de endpoint da API, escolha Regional.
7. Escolha Salvar alterações.
8. Remova a política de recursos da sua API.
9. Reimplante sua API para que as alterações sejam aplicadas.

Como você está migrando o tipo de endpoint de privado para Regional, o API Gateway altera o tipo de endereço IP para IPv4. Para obter mais informações, consulte [the section called “Tipos de endereço IP para APIs REST no API Gateway”](#).

Converter um endpoint regional em um endpoint privado

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST.
3. Crie uma política de recursos que conceda acesso a VPC ou ao endpoint da VPC. Para obter mais informações, consulte [???](#).
4. Escolha Configurações da API.
5. Na seção Detalhes da API, escolha Editar.
6. Em Tipo de endpoint de API, escolha Privado.
7. (Opcional) Para IDs de endpoint da VPC, selecione as IDs de endpoint da VPC que você deseja associar à API privada.
8. Escolha Salvar alterações.
9. Reimplante sua API para que as alterações sejam aplicadas.

Como você está migrando o tipo de endpoint de Regional para privado, o API Gateway altera o tipo de endereço IP para pilha dupla. Para obter mais informações, consulte [the section called “Tipos de endereço IP para APIs REST no API Gateway”](#).

Usar a AWS CLI para alterar um tipo de endpoint de API

O comando [update-rest-api](#) indicado abaixo atualiza uma API otimizada para borda para uma API regional:

```
aws apigateway update-rest-api \
```

```
--rest-api-id a1b2c3 \  
--patch-operations op=replace,path=/endpointConfiguration/types/EDGE,value=REGIONAL
```

A resposta bem-sucedida tem um código de status de 200 OK e uma carga semelhante ao seguinte:

```
{  
  "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"  
}
```

O comando [update-rest-api](#) indicado abaixo atualiza uma API regional para uma API otimizada para borda:

```
aws apigateway update-rest-api \  
--rest-api-id a1b2c3 \  
--patch-operations op=replace,path=/endpointConfiguration/types/REGIONAL,value=EDGE
```

Como [put-rest-api](#) é usado para atualizar as definições de API, não é aplicável à atualização de um tipo de endpoint de API.

Políticas de segurança para APIs REST no API Gateway

Uma política de segurança é uma combinação predefinida da versão mínima do TLS e dos pacotes de criptografia oferecida pelo Amazon API Gateway. Quando seus clientes estabelecem um handshake do TLS para a API ou o nome do domínio personalizado, a política de segurança aplica a versão do TLS e o pacote de criptografia aceitos pelo API Gateway. As políticas de segurança protegem suas APIs e nomes de domínio personalizados contra problemas de segurança de rede, como violação e espionagem entre um cliente e o servidor.

O API Gateway aceita políticas de segurança legadas e políticas de segurança aprimoradas. TLS_1_0 e TLS_1_2 são políticas de segurança legadas. Use essas políticas de segurança para compatibilidade com versões anteriores. Qualquer política que comece com SecurityPolicy_ é uma política de segurança aprimorada. Use essas políticas para workloads reguladas, governança avançada ou para usar criptografia pós-quântica. Ao usar uma política de segurança aprimorada,

você também deve definir o modo de acesso ao endpoint para governança adicional. Para obter mais informações, consulte [the section called “Modo de acesso ao endpoint”](#).

Como o API Gateway aplica políticas de segurança

O exemplo a seguir mostra como o API Gateway aplica políticas de segurança utilizando a política `SecurityPolicy_TLS13_1_3_2025_09` como exemplo.

A política de segurança `SecurityPolicy_TLS13_1_3_2025_09` aceita tráfego TLS 1.3 e rejeita tráfego TLS 1.2 e TLS 1.0. Para o tráfego TLS 1.3, a política de segurança aceita os seguintes pacotes de criptografia:

- `TLS_AES_128_GCM_SHA256`
- `TLS_AES_256_GCM_SHA384`
- `TLS_CHACHA20_POLY1305_SHA256`

O API Gateway não aceita nenhum outro pacote de criptografia. Por exemplo, a política de segurança rejeitaria qualquer tráfego TLS 1.3 que utilize o pacote de criptografia AES128-SHA. Para acessar mais informações sobre as versões de TLS e criptografias aceitas, consulte [the section called “Políticas de segurança aceitas”](#).

Para monitorar quais protocolos TLS e criptografias os clientes usaram para acessar seu API Gateway, você pode usar as variáveis de contexto `$context.tlsVersion` e `$context.cipherSuite` em seus logs de acesso. Para obter mais informações, consulte [the section called “Monitorar”](#).

Modo de acesso ao endpoint

O modo de acesso ao endpoint é um parâmetro adicional que você deve especificar para qualquer API REST ou nome de domínio personalizado que use uma política de segurança aprimorada começando com `SecurityPolicy_`. Você faz isso ao criar seu recurso ou ao alterar a política de segurança de uma política legada para uma política aprimorada.

Quando o modo de acesso ao endpoint é definido como `STRICT`, todas as solicitações para sua API REST ou nome de domínio personalizado devem passar pelas seguintes verificações:

- A solicitação deve ser originada do mesmo tipo de endpoint do API Gateway que seu recurso. Pode ser de um endpoint regional, otimizado para borda ou privado.

- Se você usar um endpoint regional ou privado, o API Gateway usará a correspondência de host SNI. Se você utilizar um endpoint otimizado para borda, o API Gateway estará em conformidade com a proteção contra “domain fronting” do CloudFront. Para acessar mais informações, consulte [Domain fronting](#).

Se alguma dessas condições não for atendida, o API Gateway rejeitará a solicitação. É recomendável utilizar o modo de acesso ao endpoint STRICT quando possível.

Para migrar uma API ou nome de domínio existente para usar o modo de acesso ao endpoint restrito, primeiro atualize sua política de segurança para uma política de segurança aprimorada e mantenha o modo de acesso ao endpoint definido como BASIC. Depois de validar seus logs de tráfego e acesso, defina o modo de acesso ao endpoint como STRICT. Quando você migra o modo de acesso ao endpoint de STRICT para BASIC, ele fica indisponível por cerca de 15 minutos à medida que as alterações se propagam.

Você não deve definir o modo de acesso ao endpoint como STRICT para determinadas arquiteturas de aplicações e, em vez disso, definir o modo de acesso ao endpoint como BASIC. A tabela a seguir mostra algumas arquiteturas de aplicações e uma recomendação para que sua API REST ou nome de domínio personalizado possa usar o modo de acesso ao endpoint STRICT.

Arquitetura	Migração sugerida
Usar um endpoint da VPC para acessar um nome de domínio personalizado público.	Essa arquitetura utiliza tráfego do tipo entre endpoints. Recomendamos migrar para the section called “Nomes de domínio personalizados para APIs privadas” .
Usar qualquer método para invocar uma API privada que não utilize um nome de domínio personalizado ou nomes DNS privados.	Essa arquitetura cria uma incompatibilidade entre o cabeçalho d host e o SNI utilizado no handshake TLS e não atende às restrições de “domain fronting” do CloudFront. Recomendamos que você migre sua VPC para utilizar o DNS privado.
Utilizar a fragmentação de domínio para distribuir conteúdo em vários domínios ou subdomínios.	Essa arquitetura cria uma incompatibilidade entre o cabeçalho d host e o SNI utilizado no handshake TLS e não atende às restrições de “domain fronting” do CloudFront. É recomendá

Arquitetura	Migração sugerida
	vel que você utilize HTTP/2 e migre para longe desse antipadrão.

Veja a seguir algumas considerações sobre o uso do modo de acesso ao endpoint:

- Se o modo de acesso ao endpoint de uma API ou nome de domínio for **STRICT**, você não poderá alterar o tipo de endpoint. Para alterar o tipo de endpoint, primeiro altere o modo de acesso ao endpoint para **BASIC**.
- Depois de alterar o modo de acesso ao endpoint de **BASIC** para **STRICT**, há um atraso de 15 minutos para o API Gateway aplicar o modo de acesso restrito ao endpoint.
- Ao alterar uma política de segurança de uma política que começa com `SecurityPolicy_` para uma política legada, você deve cancelar a definição do modo de acesso ao endpoint como `""`.

Considerações

Veja a seguir algumas considerações sobre políticas de segurança para APIs REST no API Gateway:

- Você pode importar a política de segurança em um arquivo de definição da OpenAPI. Para obter mais informações, consulte [x-amazon-apigateway-endpoint-access-mode](#).
- Sua API pode ser associada a um nome de domínio personalizado com uma política de segurança diferente da política da sua API. Quando você invoca esse nome de domínio personalizado, o API Gateway utiliza a política de segurança da API para negociar o handshake TLS. Se você desabilitar seu endpoint de API padrão, isso poderá afetar como os chamadores podem invocar sua API.
- Se você alterar sua política de segurança, a atualização levará cerca de 15 minutos para ser concluída. É possível monitorar o `apiStatus` de sua API. No decorrer da atualização da sua API, o `apiStatus` será **UPDATING** e, quando o processo for concluído, será **AVAILABLE**. Quando o status da sua API é **UPDATING**, você ainda pode invocá-la.
- O API Gateway aceita políticas de segurança em todas as APIs. No entanto, você só pode escolher uma política de segurança para as APIs REST. O API Gateway comporta apenas a política de segurança `TLS_1_2` para APIs HTTP ou de WebSocket.
- Você não pode atualizar a política de segurança de uma API de `TLS_1_0` para `TLS_1_2`.

- Algumas políticas de segurança comportam os pacotes de criptografia ECDSA e RSA. Se você usar esse tipo de política com um nome de domínio personalizado, os pacotes de criptografia corresponderão ao tipo de chave de certificado fornecido pelo cliente, RSA ou ECDSA. Se você usar esse tipo de política com uma API REST, os conjuntos de criptografia corresponderão aos pacotes aceitos com os tipos de certificados RSA.

Políticas de segurança aceitas

As tabelas a seguir descrevem as [políticas de segurança](#) que podem ser especificadas para cada tipo de endpoint de API REST e tipo de nome de domínio personalizado. Essas políticas permitem que você controle as conexões de entrada. O API Gateway aceita somente o TLS 1.2 na saída. Você pode atualizar a política de segurança da sua API ou nome de domínio personalizado a qualquer momento.

As políticas que contêm o FIPS no título são compatíveis com o Federal Information Processing Standard (FIPS), que é um padrão do governo dos EUA e do Canadá que especifica os requisitos de segurança para módulos criptográficos que protegem informações sensíveis. Para saber mais, consulte [Federal Information Processing Standard \(FIPS\) 140](#) na página AWS Cloud Security Compliance.

Todas as políticas do FIPS utilizam o módulo criptográfico AWS-LC validado pelo FIPS. Para saber mais, consulte a página [AWS-LC Cryptographic Module](#) no site NIST Cryptographic Module Validation Program.

As políticas que contêm PQ no título utilizam [criptografia pós-quântica \(PQC\)](#) para implementar algoritmos híbridos de troca de chaves para TLS a fim de garantir a confidencialidade do tráfego contra futuras ameaças de computação quântica.

As políticas que contêm PFS no título utilizam o [Perfect Forward Secrecy \(PFS\)](#) para garantir que as chaves da sessão não sejam comprometidas.

As políticas que contêm FIPS e PQ em seus títulos comportam esses dois recursos.

Políticas de segurança padrão

Quando você cria uma API REST ou domínio personalizado, o recurso recebe uma política de segurança padrão. A tabela a seguir mostra a política de segurança padrão para esses recursos.

Recurso	Nome da política de segurança padrão		
APIs regionais	TLS_1_0		
APIs otimizadas para bordas	TLS_1_0		
APIs privadas	TLS_1_2		
Domínios regionais	TLS_1_2		
Domínio otimizado para borda	TLS_1_2		
Domínio privado	TLS_1_2		

Políticas de segurança aceitas para APIs regionais e privadas e nomes de domínio personalizados

A tabela a seguir descreve as políticas de segurança que podem ser especificadas para nomes de domínio personalizados e APIs privadas e regionais.

Política de segurança	Versões aceitas do TLS	Criptografias compatíveis
SecurityPolicy_TLS13_1_3_2025_09	TLS1.3	TLS1.3 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256
SecurityPolicy_TLS13_1_3_FIPS_2025_09	TLS1.3	TLS1.3 • TLS_AES_128_GCM_SHA256

Política de segurança	Versões aceitas do TLS	Criptografias compatíveis
		• 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 segurança	Versões aceitas do TLS	Criptografias compatíveis
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 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384

Política de segurança	Versões aceitas do TLS	Criptografias compatíveis
SecurityPolicy_TLS 13_1_2_PQ_2025_09	TLS1.3	TLS1.3
	TLS1.2	• TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 TLS1.2 • TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • 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

Política de segurança	Versões aceitas do TLS	Criptografias compatíveis
		• AES128-SHA256 • AES256-GCM-SHA384 • AES256-SHA256

Política de segurança	Versões aceitas do TLS	Criptografias compatíveis
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 segurança	Versões aceitas do TLS	Criptografias compatíveis
TLS_1_2	TLS1.3	TLS1.3
	TLS1.2	• TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384 • AES128-GCM-SHA256 • AES128-SHA256 • AES256-GCM-SHA384 • AES256-SHA256

Política de segurança	Versões aceitas do TLS	Criptografias compatíveis
TLS_1_0	TLS1.3	TLS1.3
	TLS1.2	• TLS_AES_128_GCM_SHA256
	TLS1.1	• TLS_AES_256_GCM_SHA384
	TLS1.0	• TLS_CHACHA20_POLY1305_SHA256
		TLS1.0-TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-ECDSA-AES128-SHA • ECDHE-RSA-AES128-SHA • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES256-SHA384 • ECDHE-ECDSA-AES256-SHA

Política de segurança	Versões aceitas do TLS	Criptografias compatíveis
		• ECDHE-RSA-AES256-SHA • AES128-GCM-SHA256 • AES128-SHA256 • AES128-SHA • AES256-GCM-SHA384 • AES256-SHA256 • AES256-SHA

Políticas de segurança aceitas para APIs otimizadas para borda e nomes de domínio personalizados

A tabela a seguir descreve as políticas de segurança que podem ser especificadas para APIs e nomes de domínio personalizados otimizados para borda.

Nome da política de segurança	Versões aceitas do TLS	Criptografias compatíveis
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

Nome da política de segurança	Versões aceitas do TLS	Criptografias compatíveis
		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

Nome da política de segurança	Versões aceitas do TLS	Criptografias compatíveis
SecurityPolicy_TLS12_2018_E DGE	TLS1.3	TLS1.3
	TLS1.2	• TLS_AES_128_GCM_SHA256 • TLS_AES_256_GCM_SHA384 • TLS_CHACHA20_POLY1305_SHA256 TLS1.2 • ECDHE-ECDSA-AES128-GCM-SHA256 • ECDHE-ECDSA-AES128-SHA256 • ECDHE-ECDSA-AES256-GCM-SHA384 • ECDHE-ECDSA-CHACHA20-POLY1305 • ECDHE-ECDSA-AES256-SHA384 • ECDHE-RSA-AES128-GCM-SHA256 • ECDHE-RSA-AES128-SHA256 • ECDHE-RSA-AES256-GCM-SHA384 • ECDHE-RSA-CHACHA20-POLY1305 • ECDHE-RSA-AES256-SHA384

Nome da política de segurança	Versões aceitas do TLS	Criptografias compatíveis
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

Nome da política de segurança	Versões aceitas do TLS	Criptografias compatíveis
		• ECDHE-ECDSA-AES256-SHA • ECDHE-RSA-AES256-SHA • AES256-SHA256 • AES256-SHA

Nomes das criptografias OpenSSL e RFC

OpenSSL e IETF RFC 5246 usam nomes diferentes para as mesmas cifras. A tabela a seguir mapeia o nome do OpenSSL para o nome do RFC para cada criptograma. Para ter mais informações, consulte [ciphers](#) na documentação do OpenSSL.

Nome da criptografia OpenSSL	Nome da criptografia 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

Nome da criptografia OpenSSL	Nome da criptografia 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

Como alterar uma política de segurança

É possível alterar a política de segurança para sua API. Se você está enviando tráfego para as suas APIs utilizando o seu nome de domínio personalizado, a API e o nome de domínio personalizado não precisarão ter a mesma política de segurança. Quando você invoca esse nome de domínio personalizado, o API Gateway utiliza a política de segurança da API para negociar o handshake TLS. No entanto, para manter a consistência, é recomendável usar a mesma política de segurança para o seu nome de domínio personalizado e a API.

Se você alterar sua política de segurança, a atualização levará cerca de 15 minutos para ser concluída. É possível monitorar o `apiStatus` de sua API. No decorrer da atualização da sua API, o `apiStatus` será `UPDATING` e, quando o processo for concluído, será `AVAILABLE`. Durante a atualização da sua API, você ainda pode invocá-la.

Console de gerenciamento da AWS

Como alterar a política de segurança de uma API

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST.
3. Selecione Configurações da API e, depois, escolha Editar.
4. Em Política de segurança, selecione uma nova política que comece com `SecurityPolicy_`.
5. Em Modo de acesso ao endpoint, escolha Restrito.
6. Escolha Salvar alterações.

Implante a API novamente para que as alterações entrem em vigor. Como você alterou o modo de acesso ao endpoint para restrito, levará cerca de 15 minutos para que as alterações se propaguem totalmente.

AWS CLI

O seguinte comando [update-rest-api](#) atualiza uma API para utilizar a política de segurança `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"
    }
  ]'
```



```
}  
]'
```

A saída será exibida da seguinte forma:

```
{  
  "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"  
}
```

O comando [update-rest-api](#) a seguir atualiza uma API que estava utilizando uma política de segurança aprimorada para utilizar 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": ""  
    }  
  ]'
```

A saída será exibida da seguinte forma:

```
{
  "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 endereço IP para APIs REST no API Gateway

Ao criar uma API, você especifica o tipo de endereços IP que podem invocar sua API. É possível escolher IPv4 para resolver endereços IPv4 e invocar sua API ou escolher pilha dupla para permitir que endereços IPv4 e IPv6 invoquem sua API. Recomendamos que você defina o tipo de endereço IP como pilha dupla para aliviar o esgotamento do espaço IP ou para seu procedimento de segurança. Para ter mais informações sobre os benefícios de um tipo de endereço IP de pilha dupla, consulte [IPv6 na AWS](#).

Para restringir sua API somente ao tráfego IPv6, você pode criar uma política de recursos e restringir os endereços IP de origem somente a intervalos IPv6. É possível alterar o tipo de endereço IP atualizando a configuração da API. Essa alteração entrará em vigor imediatamente e você não precisará reimplantar sua API. Para obter mais informações, consulte [the section called “Exemplo: negar tráfego da API com base no intervalo ou endereço IP de origem”](#).

Considerações sobre tipos de endereço IP

As seguintes considerações podem afetar o uso de tipos de endereço IP:

- O tipo de endereço IP padrão para todas as APIs regionais e otimizadas para borda é IPv4.

- As APIs privadas só podem ter um tipo de endereço IP de pilha dupla.
- Se você alterar o tipo de endereço IP de uma API existente de IPv4 para pilha dupla, confirme se todas as políticas que controlam o acesso às suas APIs foram atualizadas para considerar as chamadas IPv6. Quando você altera o tipo de endereço IP, a alteração entra em vigor imediatamente.
- Se você migrar o tipo de endpoint de uma API de regional ou otimizado para borda para privado, o API Gateway alterará o tipo de endereço IP para pilha dupla. Para obter mais informações, consulte [the section called “Alterar um tipo de endpoint de API pública ou privada”](#).
- Se você migrar o tipo de endpoint de uma API de privado para regional, deverá definir o tipo de endereço IP como pilha dupla. Após a conclusão da migração do endpoint, você pode alterar o tipo de endereço IP para IPv4. Para obter mais informações, consulte [the section called “Alterar um tipo de endpoint de API pública ou privada”](#).
- Sua API pode ser associada a um nome de domínio personalizado com um tipo de endereço IP diferente do da sua API. Se você desabilitar seu endpoint de API padrão, isso poderá afetar como os chamadores podem invocar sua API.
- Não é possível usar um arquivo de definição externo para configurar o tipo de endereço IP da sua API.

Alterar o tipo de endereço IP de uma API REST

É possível alterar o tipo de endereço IP atualizando a configuração da API. É possível atualizar a configuração da API usando o Console de gerenciamento da AWS, a AWS CLI, o CloudFormation ou um SDK da AWS. Se você alterar o tipo de endereço IP da API, não poderá replantá-la para que as alterações entrem em vigor. Antes de alterar o tipo de endereço IP, confirme se todas as políticas que controlam o acesso às suas APIs foram atualizadas para considerar as chamadas IPv6.

Console de gerenciamento da AWS

Como alterar o tipo de endereço IP de uma API REST

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST.
3. Selecione Configurações da API e, depois, escolha Editar.
4. Em Tipo de endereço IP, escolha IPv4 ou Pilha dupla.
5. Escolha Salvar alterações.

A alteração na configuração da sua API entrará em vigor imediatamente.

AWS CLI

O seguinte comando [update-rest-api](#) atualiza uma API para ter um tipo de endereço IP de pilha dupla:

```
aws apigateway update-rest-api \  
  --rest-api-id abcd1234 \  
  --patch-operations "op='replace',path='/endpointConfiguration/  
ipAddressType',value='dualstack'"
```

A saída será exibida da seguinte forma:

```
{  
  "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 para APIs REST no API Gateway

No API Gateway, um método de API incorpora uma [solicitação de método](#) e uma [resposta de método](#). Você pode configurar um método de API para definir o que um cliente deve fazer para enviar uma solicitação para acessar o serviço no backend e definir as respostas que o cliente recebe em troca. Para entrada, você pode escolher os parâmetros de solicitação de método ou uma carga útil aplicável para o cliente fornecer os dados obrigatórios ou opcionais em tempo de execução. Para a saída, você determina o código de status de resposta de método, os cabeçalhos e o corpo

aplicável como destinos para mapear os dados de resposta de backend, antes de serem retornados para o cliente. Para ajudar o desenvolvedor do cliente a entender os comportamentos e os formatos de entrada e saída de sua API, você pode [documentar sua API](#) e [fornecer mensagens de erro adequadas](#) para [solicitações inválidas](#).

Solicitação de método de API é uma solicitação HTTP. Para configurar a solicitação de método, configure um método HTTP (ou verbo), o caminho para um [recurso](#) de API, cabeçalhos e parâmetros de string de consulta aplicáveis. Você também configura uma carga útil quando o método HTTP é POST, PUT ou PATCH. Por exemplo, para recuperar um animal de estimação usando a [API de exemplo PetStore](#), defina a solicitação de método de API de GET `/pets/{petId}`, em que `{petId}` é um parâmetro de caminho que pode obter um número em tempo de execução.

```
GET /pets/1
Host: apigateway.us-east-1.amazonaws.com
...
```

Se o cliente especificar um caminho incorreto, por exemplo, `/pet/1` ou `/pets/one` em vez de `/pets/1`, será lançada uma exceção.

Uma resposta de método de API é uma resposta HTTP com um código de status determinado. Para uma integração não proxy, você deve configurar respostas de método para especificar os destinos obrigatórios ou opcionais dos mapeamentos. Eles transformam os cabeçalhos de resposta de integração ou o corpo para os cabeçalhos de resposta de método associado ou o corpo. O mapeamento pode ser tão simples quanto uma [transformação de identidade](#) que transmite os cabeçalhos ou o corpo pela integração no estado em que se encontra. Por exemplo, a resposta de método 200 a seguir mostra um exemplo de passagem de uma resposta de integração bem-sucedida no estado em que se encontra.

```
200 OK
Content-Type: application/json
...

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

Em princípio, você pode definir uma resposta de método correspondente a uma resposta específica do backend. Normalmente, isso envolve qualquer resposta 2XX, 4XX e 5XX. No entanto, isso pode

não ser prático, pois, muitas vezes, você pode não saber com antecedência todas as respostas que um backend pode retornar. Na prática, você pode designar uma resposta de método como padrão para lidar com respostas desconhecidas ou não mapeadas do backend. É uma boa prática designar a resposta 500 como padrão. Em qualquer caso, você deve configurar pelo menos uma resposta de método para integrações não proxy. Caso contrário, o API Gateway retorna uma resposta de erro 500 para o cliente, mesmo quando a solicitação for bem-sucedida no backend.

Para oferecer suporte a um SDK fortemente tipado, como um Java SDK, para sua API, você deve definir o modelo de dados para entrada para a solicitação de método e definir o modelo de dados para a saída da resposta de método.

Pré-requisitos

Antes de configurar um método de API, verifique o seguinte:

- Você deve ter o método disponível no API Gateway. Siga as instruções em [Tutorial: Crie uma API REST com uma integração de não proxy de HTTP](#).
- Se você deseja que o método se comunique com uma função do Lambda, você já deve ter criado as funções de invocação e execução do Lambda no IAM. Você também deve ter criado a função do Lambda com a qual seu método se comunicará no AWS Lambda. Para criar as funções, use as instruções em [Criar uma função do Lambda para integração não proxy do Lambda](#) do [Escolher um tutorial de integração do AWS Lambda](#).
- Se quiser que o método se comunique com uma integração HTTP ou de proxy HTTP, será necessário já ter criado a URL de endpoint HTTP com a qual o seu método se comunicará, além de ter acesso a ela.
- Verifique se os certificados para os endpoints HTTP e de proxy HTTP têm suporte pelo API Gateway. Para obter mais detalhes, consulte [Autoridades de certificado compatíveis com o API Gateway para integrações HTTP e de proxy HTTP no API Gateway](#).

Tópicos

- [Configurar uma solicitação de método no API Gateway](#)
- [Configurar uma resposta de método no API Gateway](#)
- [Configurar um método usando o console do API Gateway](#)

Configurar uma solicitação de método no API Gateway

A configuração de uma solicitação de método envolve as seguintes tarefas, depois de criar um recurso [RestApi](#):

1. A criação de uma nova API ou a escolha de uma entidade [Recurso](#) de API existente.
2. A criação de um recurso [Método](#) da API que seja um verbo HTTP específico no Resource de API novo ou escolhido. Essa tarefa pode ser dividida ainda mais nas seguintes tarefas secundárias:
 - Adição de um método HTTP à solicitação de método
 - Configuração de parâmetros de solicitação
 - Definição de um modelo para o corpo de solicitação
 - Adoção de um esquema de autorização
 - Ativação da validação de solicitação

Você pode executar essas tarefas usando os seguintes métodos:

- [Console do API Gateway](#)
- Comandos da AWS CLI ([create-resource](#) e [put-method](#))
- Funções do SDK da AWS (por exemplo, em Node.js, [createResource](#) e [putMethod](#))
- API REST do API Gateway ([resource:create](#) e [method:put](#)).

Tópicos

- [Configurar recursos da API](#)
- [Configurar um método HTTP](#)
- [Configurar parâmetros da solicitação de método](#)
- [Configurar um modelo de solicitação de método](#)
- [Configurar a autorização de solicitação de método](#)
- [Configurar a validação da solicitação de método](#)

Configurar recursos da API

Em uma API do API Gateway, você expõe recursos endereçáveis como uma árvore de entidades de [Recursos](#) da API, com o recurso raiz (/) na parte superior da hierarquia. O recurso raiz é relativo à URL base da API, que é composta do endpoint de API e um nome de etapa. No console do API

Gateway esse URI base é referido como o URI de invocação e é exibido no editor de etapas da API depois que a API é implantada.

O endpoint da API pode ser um nome de host padrão ou um nome de domínio personalizado. O nome de host padrão é do seguinte formato:

```
{api-id}.execute-api.{region}.amazonaws.com
```

Neste formato, o `{api-id}` representa o identificador de API que é gerado pelo API Gateway. A variável `{region}` representa a região da AWS (por exemplo, `us-east-1`) escolhida ao criar a API. Um nome de domínio personalizado é qualquer nome amigável em um domínio de internet válido. Por exemplo, se você tiver registrado um domínio da Internet de `example.com`, qualquer `*.example.com` é um nome de domínio personalizado válido. Para obter mais informações, consulte [criar um nome de domínio personalizado](#).

Para a [API de exemplo PetStore](#), o recurso raiz (`/`) expõe a loja de animais de estimação. O recurso `/pets` representa a coleção de animais de estimação disponíveis na loja. O `/pets/{petId}` expõe uma animação de estimação individual de um determinado identificador (`petId`). O parâmetro de caminho de `{petId}` faz parte dos parâmetros da solicitação.

Para configurar um recurso de API, você escolhe um recurso existente como seu pai e, em seguida, cria os recursos filho abaixo desse recurso pai. Você começa com o recurso raiz como um pai, adiciona um recurso a esse pai, adiciona outro recurso a esse recurso filho como novo pai, e assim por diante, ao identificador pai. Em seguida, você adiciona o recurso indicado ao pai.

O comando [get-resources](#) indicado abaixo recupera todos os recursos de uma API:

```
aws apigateway get-resources --rest-api-id apiId
```

Para o exemplo de API PetStore, essa lista é semelhante à seguinte:

```
{
  "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 item lista os identificadores do recurso (`id`) e, exceto para o recurso raiz, seu pai imediato (`parentId`), bem como o nome do recurso (`pathPart`). O recurso raiz é especial, pois não tem nenhum pai. Depois de escolher um recurso como primário, use o comando a seguir para adicionar um recurso secundário.

```

aws apigateway create-resource --rest-api-id apiId \
  --parent-id parentId \
  --path-part resourceName

```

Por exemplo, para colocar rações para animais de estimação à venda no site PetStore, use o seguinte comando:

```

aws apigateway create-resource --rest-api-id a1b2c3 \
  --parent-id svzr2028x8 \
  --path-part food

```

A saída será exibida da seguinte forma:

```

{
  "path": "/food",
  "pathPart": "food",
  "id": "xdsvhp",
  "parentId": "svzr2028x8"
}

```

```
}
```

Usar um recurso de proxy para simplificar a configuração de API

À medida que os negócios crescem, o proprietário da PetStore pode decidir adicionar alimentos, brinquedos e outros itens relacionados a animais de estimação para venda. Para oferecer suporte a isso, você pode adicionar `/food`, `/toys` e outros recursos ao recurso raiz. Em cada categoria de venda, você também pode adicionar mais recursos, como `/food/{type}/{item}`, `/toys/{type}/{item}`, etc. Isso pode ser entediante. Se você decidir adicionar uma camada intermediária `{subtype}` aos caminhos de recursos para alterar a hierarquia de caminhos em `/food/{type}/{subtype}/{item}`, `/toys/{type}/{subtype}/{item}`, etc., as alterações romperão a configuração da API existente. Para evitar isso, você pode usar um [recurso de proxy](#) do API Gateway para expor um conjunto de recursos da API simultaneamente.

O API Gateway define um recurso de proxy como um espaço reservado para um recurso a ser especificado quando a solicitação é enviada. Um recurso de proxy é expresso por um parâmetro de caminho especial `{proxy+}`, geralmente conhecido como um parâmetro de caminho voraz. O sinal `+` indica os recursos filho que estão anexados a ele. O espaço reservado `/parent/{proxy+}` representa qualquer recurso que corresponda ao padrão de caminho de `/parent/*`. Você pode usar qualquer string para o nome do parâmetro de caminho voraz.

O comando [create-resource](#) indicado abaixo cria um recurso de proxy sob a raiz (`/ {proxy+}`):

```
aws apigateway create-resource --rest-api-id apiId \  
  --parent-id rootResourceId \  
  --path-part {proxy+}
```

A saída será exibida da seguinte forma:

```
{  
  "path": "/{proxy+}",  
  "pathPart": "{proxy+}",  
  "id": "234jdr",  
  "parentId": "svzr2028x8"  
}
```

Para o exemplo de API PetStore, você pode usar `/ {proxy+}` para representar o `/pets` e o `/pets/{petId}`. Esse recurso de proxy também pode fazer referência a qualquer outro recurso (existente ou a ser adicionado), como `/food/{type}/{item}`, `/toys/{type}/{item}`,

etc., ou `/food/{type}/{subtype}/{item}`, `/toys/{type}/{subtype}/{item}`, etc. O desenvolvedor de backend determina a hierarquia de recursos, e o desenvolvedor do cliente é responsável por entendê-la. O API Gateway simplesmente transfere o que o cliente enviou ao backend.

Uma API pode ter mais de um recurso de proxy. Por exemplo, os recursos de proxy a seguir são permitidos em uma API, pressupondo-se que `/parent/{proxy+}` não seja o mesmo proxy principal que `/parent/{child}/{proxy+}`.

```
{proxy+}
/parent/{proxy+}
/parent/{child}/{proxy+}
```

Quando um recurso de proxy tem irmãos não proxy, os recursos irmãos são excluídos da representação do recurso de proxy. Para os exemplos anteriores, `{proxy+}` refere-se a todos os recursos sob o recurso raiz, exceto os recursos `/parent[/*]`. Ou seja, uma solicitação de método em um recurso específico tem precedência sobre uma solicitação de método em um recurso genérico no mesmo nível da hierarquia de recursos.

A tabela abaixo mostra como o API Gateway encaminha solicitações aos recursos a seguir para o estágio prod de uma API.

```
ANY {proxy+}
GET /pets/{proxy+}
GET /pets/dog
```

Solicitação	Rota selecionada	Explicação
GET <code>https://api-id.execute-api.region.amazonaws.com/prod/pets/dog</code>	GET <code>/pets/dog</code>	A solicitação corresponde totalmente a esse recurso.
GET <code>https://api-id.execute-api.region.amazonaws.com/prod/pets/cats</code>	GET <code>/pets/{proxy+}</code>	A variável de caminho voraz <code>/pets/{proxy+}</code> captura essa solicitação.

Solicitação	Rota selecionada	Explicação
GET https:// <i>api-id</i> .execute- api. <i>region</i> .amazonaws.com/prod/animals	GET /{proxy+}	A variável de caminho voraz /{proxy+} captura essa solicitação.

Um recurso de proxy não pode ter nenhum recurso filho. Qualquer recurso de API após {proxy+} é redundante e ambíguo. Os seguintes recursos de proxy não são permitidos dentro de uma API.

```
{proxy+}/child
/parent/{proxy+}/{child}
/parent/{child}/{proxy+}/{grandchild+}
```

Configurar um método HTTP

Uma solicitação de método de API é encapsulada pelo recurso [Método](#) do API Gateway. Para configurar a solicitação de método, você deve primeiro instanciar o recurso Method, definindo pelo menos um método HTTP e um tipo de autorização no método.

Intimamente associado ao recurso de proxy, o API Gateway oferece suporte a um método HTTP de ANY. Esse método ANY representa qualquer método HTTP que deve ser fornecido em tempo de execução. Ele permite usar uma única configuração de método de API para todos os métodos HTTP com suporte de DELETE, GET, HEAD, OPTIONS, PATCH, POST e PUT.

Você também pode configurar o método ANY em um recurso de proxy. Combinando o método ANY com um recurso de proxy, você obtém uma única configuração de método de API para todos os métodos HTTP com suporte em qualquer recurso de uma API. Além disso, o backend pode evoluir sem romper a configuração da API existente.

Antes de definir um método de API, considere quem pode chamar o método. Defina o tipo de autorização de acordo com seu plano. Para acesso aberto, defina-o como NONE. Para usar permissões do IAM, defina o tipo de autorização como AWS_IAM. Para usar uma função de autorizador do Lambda, defina essa propriedade como CUSTOM. Para usar um grupo de usuários do Amazon Cognito, defina o tipo de autorização como COGNITO_USER_POOLS.

O comando [put-method](#) a seguir cria uma solicitação de método para o verbo ANY usando permissões do IAM para controlar o respectivo acesso.

```
aws apigateway put-method --rest-api-id vaz7da96z6 \  
  --resource-id 6sxx2j \  
  --http-method ANY \  
  --authorization-type AWS_IAM
```

Para criar uma solicitação de método de API com um tipo de autorização diferente, consulte [the section called “Configurar a autorização de solicitação de método”](#).

Configurar parâmetros da solicitação de método

Os parâmetros da solicitação de método são uma forma de um cliente fornecer dados de entrada ou o contexto de execução necessário para concluir a solicitação de método. Um parâmetro de método pode ser um parâmetro de caminho, um cabeçalho ou um parâmetro de string de consulta. Como parte da configuração de solicitação de método, você deve declarar os parâmetros de solicitação necessários para disponibilizá-los para o cliente. Para a integração não proxy, você pode converter esses parâmetros de solicitação em um formulário que seja compatível com o requisito de backend.

Por exemplo, para a solicitação de método GET /pets/{petId}, a variável de caminho {petId} é um parâmetro de solicitação necessário. Você pode declarar esse parâmetro de caminho ao chamar o comando put-method da AWS CLI. O comando [put-method](#) indicado abaixo cria um método com um parâmetro de caminho necessário:

```
aws apigateway put-method --rest-api-id vaz7da96z6 \  
  --resource-id rjkmth \  
  --http-method GET \  
  --authorization-type "NONE" \  
  --request-parameters method.request.path.petId=true
```

Se um parâmetro não for necessário, você poderá configurá-lo como false em request-parameters. Por exemplo, se o método GET /pets usar um parâmetro de string de consulta opcional de type e um parâmetro de cabeçalho opcional de age, você pode declará-los usando o comando [put-method](#) indicado abaixo:

```
aws apigateway put-method --rest-api-id vaz7da96z6 \  
  --resource-id 6sxx2j \  
  --http-method GET \  
  --authorization-type "NONE" \  
  --request-parameters  
method.request.querystring.type=false,method.request.header.age=false
```

Em vez desse formulário resumido, você pode usar uma string JSON para definir o valor `request-parameters`:

```
'{"method.request.querystring.type":false,"method.request.header.age":false}'
```

Com essa configuração, o cliente pode consultar animais de estimação por tipo:

```
GET /pets?type=dog
```

E pode realizar uma consulta de filhotes caninos da seguinte forma:

```
GET /pets?type=dog  
age:puppy
```

Para obter informações sobre como mapear parâmetros da solicitação de método para parâmetros de solicitação de integração, consulte [the section called “Integrações”](#).

Configurar um modelo de solicitação de método

Para um método de API que possa levar dados de entrada em uma carga útil, você pode usar um modelo. Um modelo é expresso em um [esquema JSON rascunho 4](#) e descreve a estrutura de dados do corpo da solicitação. Com um modelo, um cliente pode determinar como construir uma carga útil de solicitação de método como entrada. Mais importante, o API Gateway usa o modelo para [validar uma solicitação](#), [gerar um SDK](#) e inicializar um modelo de mapeamento para configurar a integração no console do API Gateway. Para ter informações sobre como criar um [modelo](#), consulte [Noções básicas dos modelos de dados](#).

Dependendo dos tipos de conteúdo, uma carga útil de método pode ter diferentes formatos.

Um modelo é indexado no tipo de mídia da carga útil aplicada. O API Gateway usa o cabeçalho `Content-Type` da solicitação para determinar o tipo de conteúdo. Para configurar modelos de solicitação de método, adicione pares de chave-valor no formato `"media-type": "model-name"` ao mapa `requestModels` ao chamar o comando AWS CLI da `put-method`.

Para usar o mesmo modelo, independentemente do tipo de conteúdo, especifique `$default` como a chave.

Por exemplo, para definir um modelo na carga útil JSON da solicitação de método `POST /pets` da API de exemplo PetStore, você pode usar o comando [put-method](#) indicado abaixo:

```
aws apigateway put-method \
```

```
--rest-api-id vaz7da96z6 \  
--resource-id 6sxx2j \  
--http-method POST \  
--authorization-type "NONE" \  
--request-models '{"application/json":"petModel"}'
```

Aqui, `petModel` é o valor de propriedade `name` de um recurso [Model](#) que descreve um animal de estimação. A definição de esquema JSON real é expressa como um valor de string JSON da propriedade [schema](#) do recurso `Model`.

Em um Java ou outro SDK fortemente tipado, da API, os dados de entrada são emitidos como a classe `petModel` derivada da definição de esquema. Com o modelo de solicitação, os dados de entrada no SDK gerado são convertidos na classe `Empty`, que é derivada do modelo `Empty` padrão. Nesse caso, o cliente não pode instanciar a classe de dados correta para fornecer a entrada necessária.

Configurar a autorização de solicitação de método

Para controlar quem pode chamar o método de API, você pode configurar o [tipo de autorização](#) no método. Você pode usar esse tipo para adotar um dos autorizadores com suporte, incluindo funções e políticas do IAM (`AWS_IAM`), um grupo de usuários do Amazon Cognito (`COGNITO_USER_POOLS`) ou um autorizador do Lambda (`CUSTOM`).

Para usar permissões do IAM para autorizar o acesso ao método de API, defina a propriedade de entrada `authorization-type` como **`AWS_IAM`**. Ao definir essa opção, o API Gateway verifica a assinatura do autor da chamada na solicitação, com base nas credenciais do autor da chamada. Se o usuário verificado tiver permissão para chamar o método, ela será aceita. Caso contrário, a solicitação será rejeitada, e o autor da chamada receberá uma resposta de erro não autorizada. A chamada para o método não é bem-sucedida, a menos que o autor da chamada tenha permissão para invocar o método da API. A seguinte política do IAM concede permissão ao autor da chamada para chamar qualquer método de API criado na mesma Conta da AWS:

JSON

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

```
    "Action": [
      "execute-api:Invoke"
    ],
    "Resource": "arn:aws:execute-api:*:*:*"
  }
}
```

Para obter mais informações, consulte [the section called “Usar permissões do IAM”](#).

Atualmente, você só pode conceder essa política aos usuários, grupos e funções na Conta da AWS do proprietário da API. Os usuários de outra Conta da AWS poderão chamar os métodos de API somente se eles tiverem permissão para assumir uma função na Conta da AWS do proprietário da API com as permissões necessárias para chamar a ação `execute-api:Invoke`. Para obter informações sobre as permissões entre contas, consulte [Uso de funções do IAM](#).

Você pode usar a AWS CLI, um AWS SDK ou um cliente de API REST, como o [Postman](#), que implementa a [assinatura do Signature Version 4 \(SigV4\)](#).

Para usar um autorizador do Lambda para autorizar o acesso ao método de API, defina a propriedade de entrada `authorization-type` como `CUSTOM` e defina a propriedade de entrada [authorizer-id](#) como o valor da propriedade `id` de um autorizador do Lambda que já existe. O autorizador do Lambda referenciado pode ser do tipo `TOKEN` ou `REQUEST`. Para obter mais informações sobre como criar um autorizador do Lambda, consulte [the section called “Usar os autorizadores do Lambda”](#).

Para usar um grupo de usuários do Amazon Cognito para autorizar o acesso ao método de API, defina a propriedade de entrada `authorization-type` como `COGNITO_USER_POOLS` e defina a propriedade de entrada [authorizer-id](#) como o valor da propriedade `id` do autorizador `COGNITO_USER_POOLS` que já foi criado. Para obter informações sobre como criar um autorizador do grupo de usuários do Amazon Cognito, consulte [the section called “Use o grupo de usuários do Amazon Cognito como um autorizador para uma API REST”](#).

Configurar a validação da solicitação de método

Você pode habilitar a validação de solicitação ao configurar uma solicitação de método de API. Primeiro, é necessário criar um [validador de solicitação](#). O comando [create-request-validator](#) a seguir cria um validador somente do corpo da solicitação.

```
aws apigateway create-request-validator \
```



```
--rest-api-id 7zw9uyk9kl \  
--name bodyOnlyValidator \  
--validate-request-body \  
--no-validate-request-parameters
```

A saída será exibida da seguinte forma:

```
{  
  "validateRequestParameters": false,  
  "validateRequestBody": true,  
  "id": "jgppy6",  
  "name": "bodyOnlyValidator"  
}
```

Você pode usar esse validador de solicitação para utilizar a validação de solicitação como parte da configuração de solicitação de método: O comando [put-method](#) indicado abaixo cria uma solicitação de método que exige que o corpo da solicitação de entrada corresponda ao `PetModel` e tem dois parâmetros de solicitação que não são obrigatórios:

```
aws apigateway put-method \  
  --rest-api-id 7zw9uyk9kl \  
  --resource-id xdsvhp \  
  --http-method PUT \  
  --authorization-type "NONE" \  
  --request-parameters '{"method.request.querystring.type": false,  
"method.request.querystring.page":false}' \  
  --request-models '{"application/json":"petModel"}' \  
  --request-validator-id jgppy6
```

Para incluir um parâmetro de solicitação na respectiva validação, você deve definir `validateRequestParameters` como `true` para o validador da solicitação e definir o parâmetro específico da solicitação como `true` no comando `put-method`.

Configurar uma resposta de método no API Gateway

Uma resposta de método de API encapsula a saída de uma solicitação de método de API que será recebida pelo cliente. Os dados de saída incluem um código de status HTTP, alguns cabeçalhos e possivelmente um corpo.

Com integrações não proxy, os parâmetros de resposta especificados e o corpo podem ser mapeados dos dados de resposta de integração associados ou podem receber certos valores

estáticos de acordo com os mapeamentos. Esses mapeamentos são especificados na resposta de integração. O mapeamento pode ser uma transformação idêntica que passe pela resposta de integração no estado em que se encontra.

Com uma integração de proxy, o API Gateway passa a resposta de backend para a resposta de método automaticamente. Não há necessidade de configurar a resposta de método de API. No entanto, com a integração do proxy do Lambda, a função do Lambda deve retornar um resultado [desse formato de saída](#) para o API Gateway mapear com êxito a resposta de integração a uma resposta de método.

Programaticamente, a configuração da resposta de método equivale à criação de um recurso [MethodResponse](#) do API Gateway e à definição das propriedades de [statusCode](#), [responseParameters](#) e [responseModels](#).

Ao definir os códigos de status de um método de API, você deve escolher um como o padrão para lidar com qualquer resposta de integração de um código de status imprevisto. É razoável definir 500 como padrão, pois isso equivale à geração de respostas não mapeadas de outra forma como um erro no servidor. Por motivos educacionais, o console do API Gateway define a resposta 200 como padrão. No entanto, você pode redefini-la como a resposta 500.

Para configurar uma resposta de método, você deve ter criado a solicitação de método.

Configurar o código de status de resposta de método

O código de status de uma resposta de método define um tipo de resposta. Por exemplo, as respostas 200, 400 e 500 indicam respostas bem-sucedidas de erros no servidor e no cliente, respectivamente.

Para configurar um código de status de resposta de método, defina a propriedade [statusCode](#) como um código de status HTTP. O comando [put-method-response](#) a seguir cria uma resposta de método 200.

```
aws apigateway put-method-response \  
  --rest-api-id vaz7da96z6 \  
  --resource-id 6sxx2j \  
  --http-method GET \  
  --status-code 200
```

Configurar parâmetros da resposta de método

Os parâmetros da resposta de método definem quais cabeçalhos o cliente recebe em resposta à solicitação de método associada. Os parâmetros de resposta também especificam um destino para o qual o API Gateway mapeia um parâmetro de resposta de integração de acordo com os mapeamentos prescritos na resposta de integração do método de API.

Para configurar os parâmetros de resposta de método, adicione ao mapa `responseParameters` de pares de chave-valor `MethodResponse` do formato `"{parameter-name}": "{boolean}"`. O comando `put-method-response` a seguir define o cabeçalho `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 resposta de método

Um modelo de resposta de método define um formato do corpo de resposta de método. A configuração de um modelo de resposta de método é necessária quando você gera um SDK fortemente tipado para a API. Ela garante que a saída seja gerada em uma classe apropriada em Java ou Objective-C. Em outros casos, a definição de um modelo é opcional.

Antes de configurar o modelo de resposta, você deve primeiro criar o modelo no API Gateway. Para fazer isso, você pode chamar o comando [create-model](#). O comando [create-model](#) a seguir cria o modelo PetStorePet para descrever o corpo da resposta à solicitação do 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" }, \  
      "name": { "type": "string" }, \  
      "category": { "type": "string" }, \  
      "photoUrls": { "type": "array", "items": { "type": "string" } }, \  
      "tags": { "type": "array", "items": { "type": "string" } }, \  
      "status": { "type": "string" } \  
    } \  
  }'
```

```
        "type": { "type": "string" }, \
        "price": { "type": "number" } \
    } \
}'
```

O resultado é criado como um recurso [Model](#) do API Gateway.

Para configurar os modelos de resposta de método a fim de definir o formato de carga útil, adicione o par de chave-valor "application/json":"PetStorePet" ao mapa [requestModels](#) do recurso [MethodResponse](#). O comando [put-method-response](#) indicado abaixo cria uma resposta de método que usa um modelo de resposta para definir o formato da carga útil:

```
aws apigateway put-method-response \
  --rest-api-id vaz7da96z6 \
  --resource-id 6sxx2j \
  --http-method GET \
  --status-code 200 \
  --response-parameters method.response.header.my-header=false \
  --response-models '{"application/json":"PetStorePet"}'
```

Configurar um método usando o console do API Gateway

Ao criar um método usando o console da API REST, você configura a solicitação de integração e a solicitação de método. Por padrão, o API Gateway cria a resposta do método 200 para seu método.

As instruções a seguir mostram como editar as configurações de solicitação de método e como criar respostas de método adicionais para seu método.

Tópicos

- [Editar uma solicitação de método do API Gateway no console do API Gateway](#)
- [Configurar uma resposta do método API Gateway usando o console do API Gateway](#)

Editar uma solicitação de método do API Gateway no console do API Gateway

Estas instruções supõem que você já criou sua solicitação de método. Para saber mais sobre como criar um método, consulte [the section called “ Configurar uma solicitação de integração usando o console”](#).

1. No painel Recursos, escolha o método e selecione a guia Solicitação de método.

2. Na seção Configurações de solicitação de método, selecione Editar.
3. Em Autorização, selecione um autorizador disponível.
 - a. Para habilitar o acesso aberto ao método para qualquer usuário, selecione Nenhum. Essa etapa poderá ser ignorada se a configuração padrão não tiver sido alterada.
 - b. Para usar permissões do IAM para controlar o acesso do cliente ao método, selecione AWS_IAM. Com essa opção, somente os usuários das funções do IAM com a política do IAM correta anexada terão permissão para chamar esse método.

Para criar a função do IAM, especifique uma política de acesso com um formato semelhante ao seguinte:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:us-east-1:111111111111:aaabbb/*/GET/"
      ]
    }
  ]
}
```

Nesta política de acesso, *arn:aws:execute-api:us-east-1:111111111111:aaabbb*/*/GET/ é o ARN do seu método. Você pode encontrar o ARN do seu método selecionando o método na página Recursos. Para obter mais informações sobre como definir as permissões do IAM, consulte [Controlar o acesso a uma API REST com permissões do IAM](#).

Para criar um perfil do IAM, é possível adaptar as instruções no seguinte tutorial: [???](#).

- c. Para usar um autorizador do Lambda, selecione um autorizador de solicitação ou token. Crie um autorizador do Lambda para que essa opção seja exibida no menu suspenso.

Para obter informações sobre como criar um autorizador do Lambda, consulte [Usar os autorizadores do API Gateway Lambda](#).

- d. Para usar um grupo de usuários do Amazon Cognito, escolha um grupo de usuários disponíveis em Cognito user pool authorizers (Autorizadores do grupo de usuários do Cognito). Crie um grupo de usuários no Amazon Cognito e um autorizador do grupo de usuários do Amazon Cognito no API Gateway para que essa opção seja exibida no menu suspenso. Para obter informações sobre como criar um autorizador do grupo de usuários do Amazon Cognito, consulte [Controlar o acesso a APIs REST usando grupos de usuários do Amazon Cognito como autorizador](#).
4. Para especificar a validação da solicitação, selecione um valor no menu suspenso Validador de solicitação. Para desativar a validação da solicitação, selecione Nenhuma. Para obter mais informações sobre cada opção, consulte [Solicitar validação para APIs REST no API Gateway](#).
5. Selecione Chave de API obrigatória para exigir uma chave de API. Quando ativadas, as chaves de APIs são usadas em [planos de uso](#) para controlar o tráfego do cliente.
6. (Opcional) Para atribuir um nome da operação em um SDK do Java dessa API, gerada pelo API Gateway, em Nome da operação, digite um nome. Por exemplo, para a solicitação de método de GET /pets/{petId}, o nome da operação de Java SDK correspondente é, por padrão, GetPetsPetId. Esse nome é construído a partir do verbo HTTP do método (GET) e os nomes de variáveis de caminho de recurso (Pets e PetId). Se você definir o nome da operação como getPetById, o nome da operação de SDK se tornará GetPetById.
7. Para adicionar um parâmetro de string de consulta ao método, faça o seguinte:
 - a. Selecione Parâmetros de string de consulta de URL e escolha Adicionar string de consulta.
 - b. Em Nome, digite o nome do parâmetro de string de consulta.
 - c. Selecione Obrigatório se o parâmetro de string de consulta recém-criado precisar ser usado para a validação de solicitação. Para obter mais informações sobre a validação de solicitação, consulte [Solicitar validação para APIs REST no API Gateway](#).
 - d. Selecione Armazenamento em cache se o parâmetro de string de consulta recém-criado precisar ser usado como parte de uma chave de armazenamento em cache. Para obter mais informações sobre armazenamento em cache, consulte [Usar parâmetros de método ou integração como chaves de cache para indexar respostas em cache](#).

Para remover o parâmetro de string de consulta, selecione Remover.

8. Para adicionar um parâmetro de cabeçalho ao método, faça o seguinte:

- a. Selecione Cabeçalhos de solicitação HTTP e, depois, Adicionar cabeçalho.
- b. Em Nome, insira o nome do cabeçalho.
- c. Selecione Obrigatório se o cabeçalho recém-criado precisar ser usado para a validação de solicitação. Para obter mais informações sobre a validação de solicitação, consulte [Solicitar validação para APIs REST no API Gateway](#).
- d. Selecione Armazenamento em cache se o cabeçalho recém-criado precisar ser usado como parte de uma chave de armazenamento em cache. Para obter mais informações sobre armazenamento em cache, consulte [Usar parâmetros de método ou integração como chaves de cache para indexar respostas em cache](#).

Para remover o cabeçalho, selecione Remover.

9. Para declarar o formato de carga útil de uma solicitação de método com o verbo HTTP POST, PUT ou PATCH, expanda Corpo da solicitação e faça o seguinte:
 - a. Escolha Add model (Adicionar modelo).
 - b. Em Tipo de conteúdo, insira um tipo MIME (por exemplo, `application/json`).
 - c. Em Modelo, selecione um modelo no menu suspenso. Os modelos atualmente disponíveis para a API incluem os modelos padrão Empty e Error, bem como quaisquer modelos que você tenha criado e adicionado à coleção de [Modelos](#) da API. Para obter mais informações sobre a criação de um modelo, consulte [Modelos de dados para APIs REST](#).

 Note

O modelo é útil para informar ao cliente o formato de dados esperado de uma carga útil. Ele é útil para gerar um modelo de mapeamento de esqueleto. É importante gerar um SDK altamente tipado da API em linguagens como Java, C #, Objective-C e Swift. Ele só será necessário se a validação de solicitação estiver ativada para a carga útil.

10. Escolha Salvar.

Configurar uma resposta do método API Gateway usando o console do API Gateway

Um método de API pode ter uma ou mais respostas. Cada resposta é indexada por seu código de status HTTP. Por padrão, o console do API Gateway adiciona a resposta 200 às respostas de

método. Você pode modificá-la, por exemplo, para que o método retorne 201 em vez disso. Você pode adicionar outras respostas, por exemplo, 409 para negação de acesso e 500 para variáveis de estágio não inicializadas utilizadas.

Para usar o console do API Gateway para modificar, excluir ou adicionar uma resposta a um método de API, siga estas instruções.

1. No painel Recursos, escolha o método e selecione a guia Resposta do método. Talvez seja necessário selecionar o botão de seta para a direita para mostrar a guia.
2. Na seção Configurações de resposta do método, selecione Criar resposta.
3. Em Código de status HTTP, insira um código de status HTTP, como 200, 400 ou 500.

Quando uma resposta retornada pelo backend não tiver uma resposta de método correspondente definida, o API Gateway não retornará a resposta para o cliente. Em vez disso, ele retornará uma resposta de erro 500 `Internal server error`.

4. Escolha Add header (Adicionar cabeçalho).
5. Em Nome de cabeçalho, insira um nome.

Para retornar um cabeçalho do back-end para o cliente, adicione o cabeçalho à resposta do método.

6. Selecione Adicionar modelo para definir um formato do corpo de resposta de método.

Digite o tipo de mídia da carga útil de resposta para Tipo de conteúdo e selecione um modelo no menu suspenso Modelos.

7. Escolha Salvar.

Para modificar uma resposta existente, acesse a resposta do método e selecione Editar. Para alterar o Código de status HTTP, selecione Excluir e crie uma resposta de método.

Para cada resposta retornada do backend, você deve ter uma resposta compatível configurada como a resposta de método. No entanto, a configuração de cabeçalhos de resposta de método e o modelo de carga útil são opcionais, a menos que você mapeie o resultado do backend para a resposta de método antes de retornar para o cliente. Além disso, um modelo de carga útil de resposta de método é importante se você pretende gerar um SDK fortemente tipado para sua API.

Controlar e gerenciar o acesso a APIs REST no API Gateway

O API Gateway oferece suporte a vários mecanismos de controle e gerenciamento de acesso à sua API.

Os mecanismos a seguir podem ser usados para autenticação e autorização:

- As políticas de recursos permitem que você crie políticas baseadas em recursos para permitir ou negar acesso a APIs e métodos de endereços IP de origem ou endpoints da VPC especificados. Para obter mais informações, consulte [the section called “Usar políticas de recursos do API Gateway”](#).
- As funções e políticas padrão do AWS IAM oferecem controles de acesso flexíveis e robustos que podem ser aplicados a uma API inteira ou a métodos individuais. As funções e políticas do IAM podem ser usadas para controlar quem pode criar, gerenciar e chamar suas APIs. Para obter mais informações, consulte [the section called “Usar permissões do IAM”](#).
- As tags do IAM podem ser usadas com as políticas do IAM para controlar o acesso. Para obter mais informações, consulte [the section called “Controle de acesso baseado em atributos”](#).
- As políticas de endpoint para VPC endpoints de interface permitem anexar políticas de recurso do IAM a endpoints da VPC de interface a fim de melhorar a segurança das [APIs privadas](#). Para obter mais informações, consulte [the section called “Usar políticas de VPC endpoint para APIs privadas”](#).
- Autorizadores do Lambda são funções do Lambda que controlam o acesso aos métodos da API REST usando a autenticação de token de portador, bem como informações descritas pelos cabeçalhos, caminhos, strings de consulta, variáveis de estágio ou parâmetros de solicitação de variáveis de contexto. Autorizadores do Lambda são usados para controlar quem pode chamar métodos da API REST. Para obter mais informações, consulte [the section called “Usar os autorizadores do Lambda”](#).
- Os grupos de usuários do Amazon Cognito permitem criar soluções de autenticação e autorização personalizáveis para suas APIs REST. Os grupos de usuários do Amazon Cognito são usados para controlar quem pode invocar métodos da API REST. Para obter mais informações, consulte [the section called “Use o grupo de usuários do Amazon Cognito como um autorizador para uma API REST”](#).

É possível usar os mecanismos a seguir para executar outras tarefas relacionadas a controle de acesso:

- O compartilhamento de recursos entre origens (CORS) permite que você controle como a API REST responde a solicitações de recursos entre domínios. Para obter mais informações, consulte [the section called “CORS”](#).
- Os certificados SSL no lado do cliente podem ser usados para verificar se as solicitações HTTP para seu sistema backend provêm do API Gateway. Para obter mais informações, consulte [the section called “Certificados do cliente”](#).
- AWS WAF O pode ser usado para proteger sua API do API Gateway contra explorações comuns da Web. Para obter mais informações, consulte [the section called “AWS WAF”](#).

É possível usar os mecanismos a seguir para rastrear e limitar o acesso que você concedeu a clientes autorizados:

- Os planos de uso permitem que você forneça chaves da API aos clientes e monitore e restrinja o uso dos estágios e métodos da API para cada chave da API. Para obter mais informações, consulte [the section called “Planos de uso”](#).

Controlar o acesso a uma API REST com políticas de recursos do API Gateway

As políticas de recursos do Amazon API Gateway são documentos de política JSON que você anexa a uma API para controlar se uma entidade principal especificada (geralmente um usuário ou um perfil do IAM) pode invocar a API. Você pode usar as políticas de recursos do API Gateway para permitir que sua API seja invocada de forma segura por:

- Usuários de uma determinada conta da AWS.
- Intervalos de endereços IP ou blocos CIDR de origem especificados.
- Nuvens privadas virtuais (VPCs) ou endpoints da VPC (em qualquer conta) específicos

Você pode anexar uma política de recursos para qualquer tipo de endpoint de API no API Gateway usando o Console de gerenciamento da AWS, a AWS CLI ou os AWS SDKs. Para [APIs privadas](#), é possível usar políticas de recurso em conjunto com políticas de VPC endpoint para controlar quais principais têm acesso a quais recursos e ações. Para obter mais informações, consulte [the section called “Usar políticas de VPC endpoint para APIs privadas”](#).

As políticas de recursos do API Gateway são diferentes das políticas baseadas em identidade do IAM. As políticas baseadas em identidade do IAM são anexadas usuários, grupos ou funções do IAM e definem quais ações essas entidades são capazes de executar em quais recursos. As políticas

de recursos do API Gateway são anexadas aos recursos. Você pode usar as políticas de recursos do API Gateway junto com as políticas do IAM. Para obter mais informações, consulte [Políticas baseadas em identidade e políticas baseadas em recurso](#).

Tópicos

- [Visão geral da linguagem de políticas de acesso para o Amazon API Gateway](#)
- [Como as políticas de recursos do API Gateway afetam o fluxo de trabalho de autorização](#)
- [Exemplos de política de recursos do API Gateway](#)
- [Criar e anexar uma política de recursos do API Gateway a uma API](#)
- [AWSAs chaves condições da que podem ser usadas nas políticas de recursos do API Gateway](#)

Visão geral da linguagem de políticas de acesso para o Amazon API Gateway

Esta página descreve os elementos básicos usados nas políticas de recursos do Amazon API Gateway.

As políticas de recursos são especificadas usando a mesma sintaxe que as políticas do IAM. Para obter as informações completas sobre a linguagem da política, consulte [Visão geral de políticas do IAM](#) e [Referência de políticas do AWS Identity and Access Management](#) no Guia do usuário do IAM.

Para obter informações sobre como um serviço da AWS determina se uma solicitação deve ser permitida ou negada, consulte [Determinar se uma solicitação é permitida ou negada](#).

Elementos comuns em uma política de acesso

No sentido mais básico, uma política de recursos contém os seguintes elementos:

- Recursos – APIs são os recursos do Amazon API Gateway para os quais você pode permitir ou negar permissões. Em uma política, você usa o nome de recurso da Amazon (ARN) para identificar o recurso. Também é possível usar a sintaxe abreviada, que o API Gateway expande automaticamente para o ARN completo ao salvar uma política de recursos. Para saber mais, consulte [Exemplos de política de recursos do API Gateway](#).

Para o formato do elemento Resource completo, consulte [Formato de recurso das permissões para executar a API no API Gateway](#).

- Ações — para cada recurso, o Amazon API Gateway oferece suporte a um conjunto de operações. Você identifica as operações de recursos que permitirão (ou negarão) usando palavras-chave de ação.

Por exemplo, a `execute-api:Invoke` permissão permite que o usuário permissão para chamar uma API mediante a solicitação de um cliente.

Para o formato do elemento `Action`, consulte [Formato de ação das permissões para executar a API no API Gateway](#).

- **Efeito** — qual é o efeito quando o usuário solicita a ação específica, que pode ser `Allow` ou `Deny`. Você também pode negar explicitamente o acesso a um recurso, o que pode fazer para ter a certeza de que um usuário não consiga acessá-lo, mesmo que uma política diferente conceda acesso.

 Note

"Negação explícita" é a mesma coisa que "Negação por padrão".
Uma "negação implícita" é diferente de uma "negação explícita". Para obter mais informações, consulte [A diferença entre negação por padrão e negação explícita](#).

- **Entidade principal**: a conta ou o usuário que tem permissão de acesso a ações e recursos na instrução. Em uma política de recursos, a entidade principal é o usuário ou a conta que recebe essa permissão.

O exemplo de política de recursos a seguir mostra os elementos comuns de política anteriores. A política concede acesso a todas as APIs no `account-id` especificado na `region` especificada para qualquer usuário cujo endereço IP de origem esteja no bloco de endereço `123.4.5.6/24`. A política nega todo o acesso à API se o IP de origem do usuário não estiver dentro do intervalo.

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"
      }
    }
  }
]
```

Como as políticas de recursos do API Gateway afetam o fluxo de trabalho de autorização

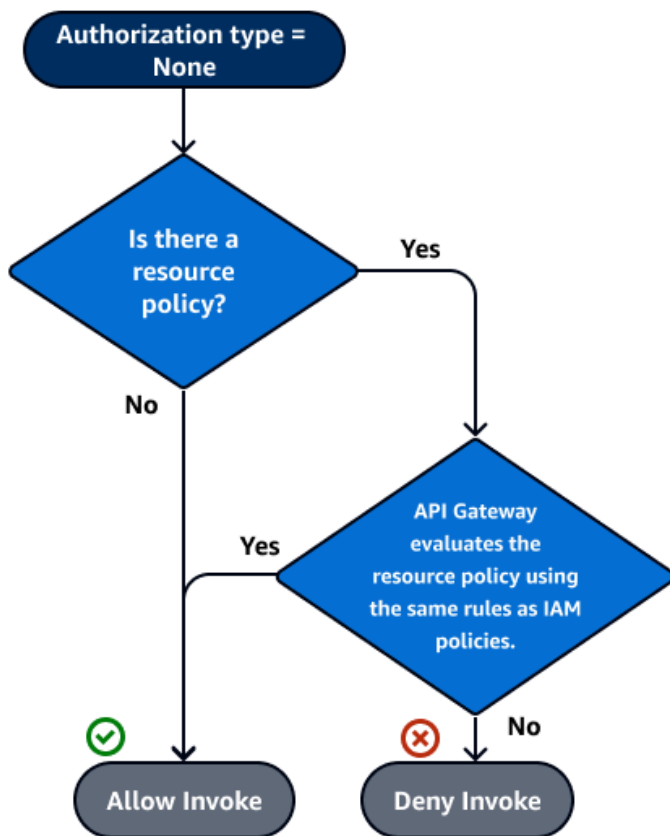
Quando o API Gateway avalia a política de recurso anexada à sua API, o resultado é afetado pelo tipo de autenticação definido para a API, conforme ilustrado nos fluxogramas das próximas seções.

Tópicos

- [Somente política de recursos do API Gateway](#)
- [Política de recursos e autorizador do Lambda](#)
- [Política de recursos e autenticação do IAM](#)
- [Autenticação e política de recursos do Amazon Cognito](#)
- [Tabelas de resultados de avaliação de política](#)

Somente política de recursos do API Gateway

Neste fluxo de trabalho, uma política de recursos do API Gateway é anexada à API, mas nenhum tipo de autenticação é definido para a API. A avaliação da política envolve a busca de uma permissão explícita baseada nos critérios de entrada do autor da chamada. Uma negação implícita ou qualquer negação explícita resulta na negação do autor da chamada.



Veja a seguir um exemplo dessa política de recursos.

JSON

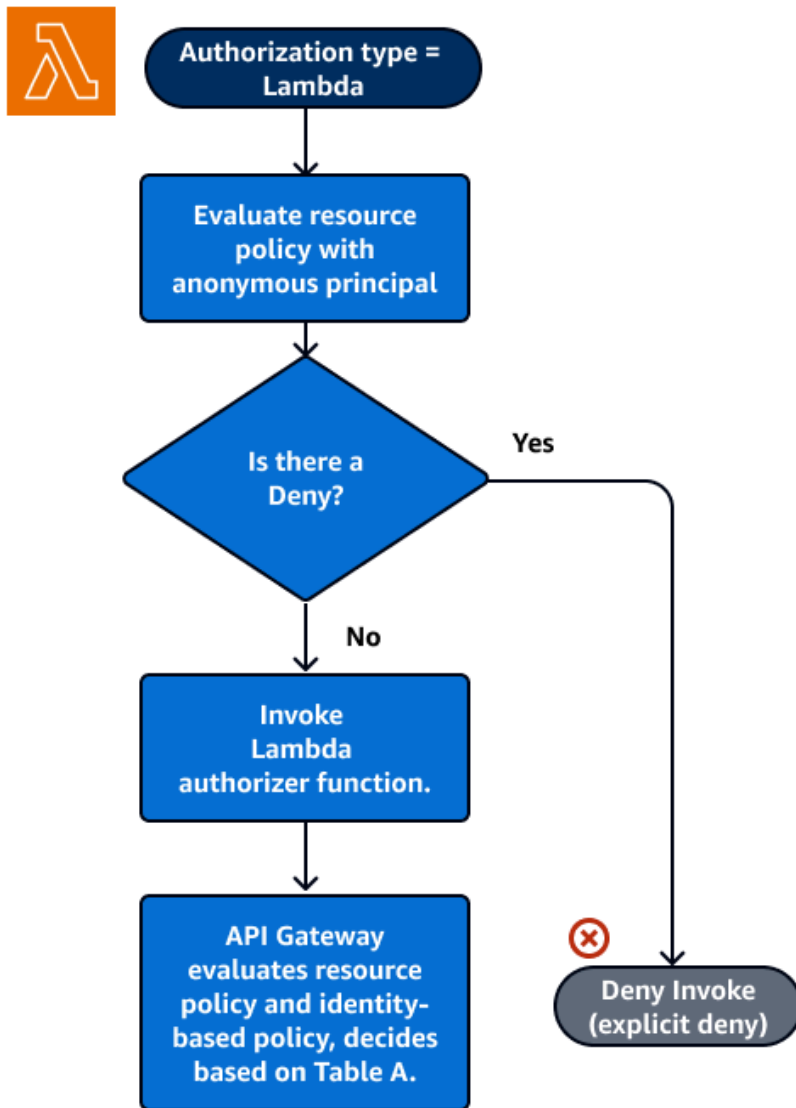
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "execute-api:Invoke",
      "Resource": "arn:aws:execute-api:us-east-1:111111111111:api-id/",
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": ["192.0.2.0/24", "198.51.100.0/24"]
        }
      }
    }
  ]
}
```

```
}
```

Política de recursos e autorizador do Lambda

Neste fluxo de trabalho, um autorizador do Lambda é configurado para a API, além de uma política de recursos. A política de recursos é avaliada em duas fases. Antes de chamar o autorizador do Lambda, o API Gateway primeiro avalia a política e verifica se há negações explícitas. Se encontradas, o autor da chamada terá o acesso negado imediatamente. Caso contrário, o autorizador do Lambda é chamado e retorna um [documento de política](#), que é avaliado em conjunto com a política de recursos. Se seu autorizador usa armazenamento em cache, o API Gateway pode exibir o documento de política em cache. O resultado é determinado com base na [Tabela A](#).

O exemplo de política de recursos a seguir permite chamadas somente a partir do VPC endpoint cujo ID de VPC endpoint é *vpce-1a2b3c4d*. Durante a avaliação de “pré-autorização”, somente as chamadas vindas do VPC endpoint indicado no exemplo são permitidas para prosseguir e avaliar o autorizador do Lambda. Todas as chamadas restantes são bloqueadas. Esse fluxo de trabalho de autorização será o mesmo se você usar um nome de domínio personalizado para uma 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": {
```



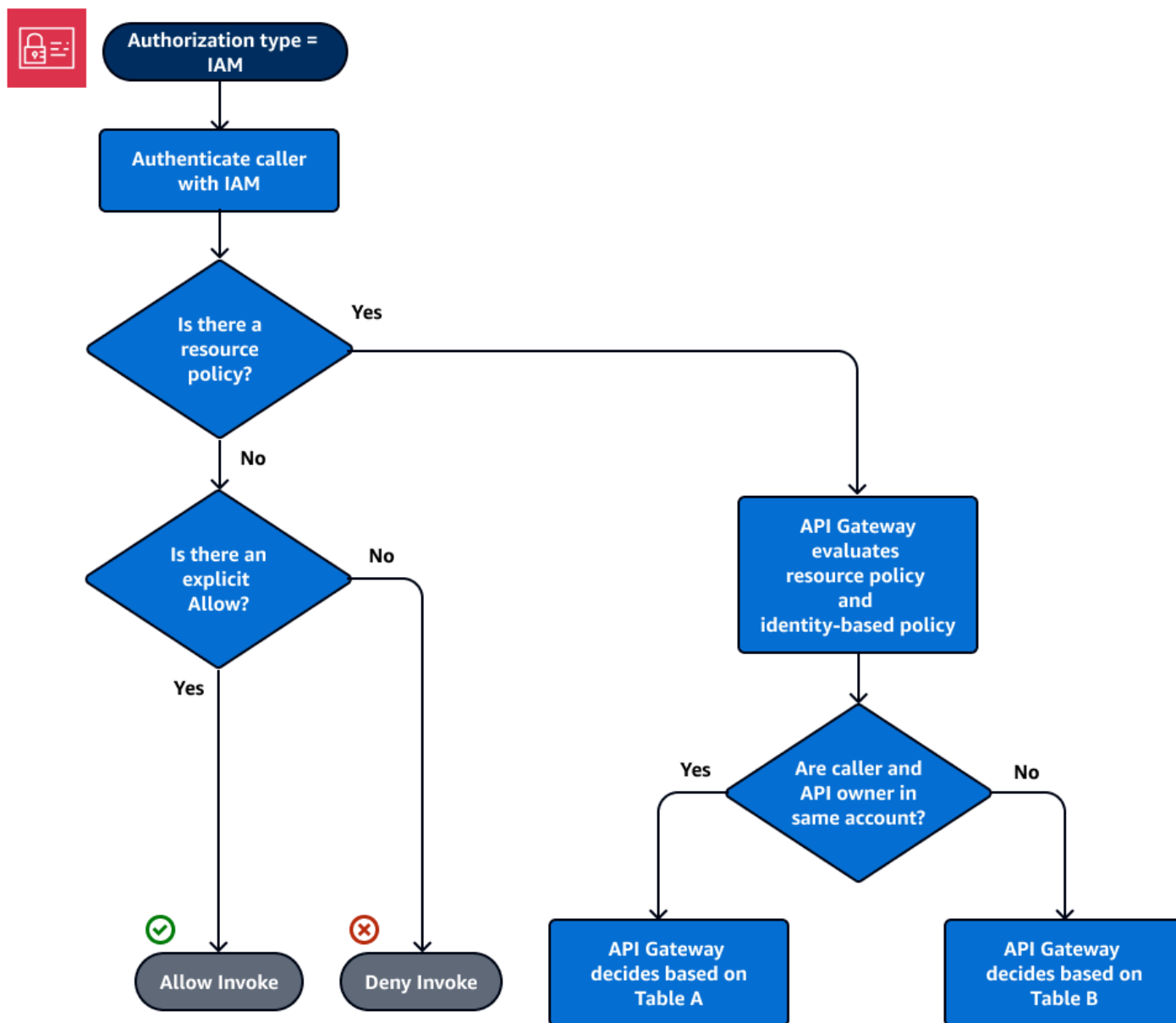
```
    "StringNotEquals": {  
      "aws:SourceVpce": "vpce-1a2b3c4d"  
    }  
  }  
]  
}
```

Política de recursos e autenticação do IAM

Nesse fluxo de trabalho, uma autenticação do IAM é configurada para a API além de uma política de recursos. Depois de autenticar o usuário com o serviço do IAM, a API avalia as duas políticas anexadas ao usuário, além da política de recursos. O resultado varia com base na origem do autor da chamada, se ele está na mesma Conta da AWS ou em outra Conta da AWS do proprietário da API.

Se o autor da chamada e o proprietário da API forem de contas diferentes, as políticas do IAM e a política de recursos deverão permitir explicitamente que o autor da chamada prossiga. Consulte mais informações em [Tabela B](#).

No entanto, se o autor da chamada e o proprietário da API estiverem na mesma Conta da AWS, as políticas de usuário do IAM ou a política de recursos deverá permitir explicitamente que o autor da chamada prossiga. Consulte mais informações em [Tabela A](#).



Veja a seguir um exemplo de uma política de recursos entre contas. Pressupondo-se que a política do IAM contenha um efeito de permissão, essa política de recursos permite chamadas somente da VPC cujo ID é *vpc-2f09a348*. Consulte mais informações em [Tabela B](#).

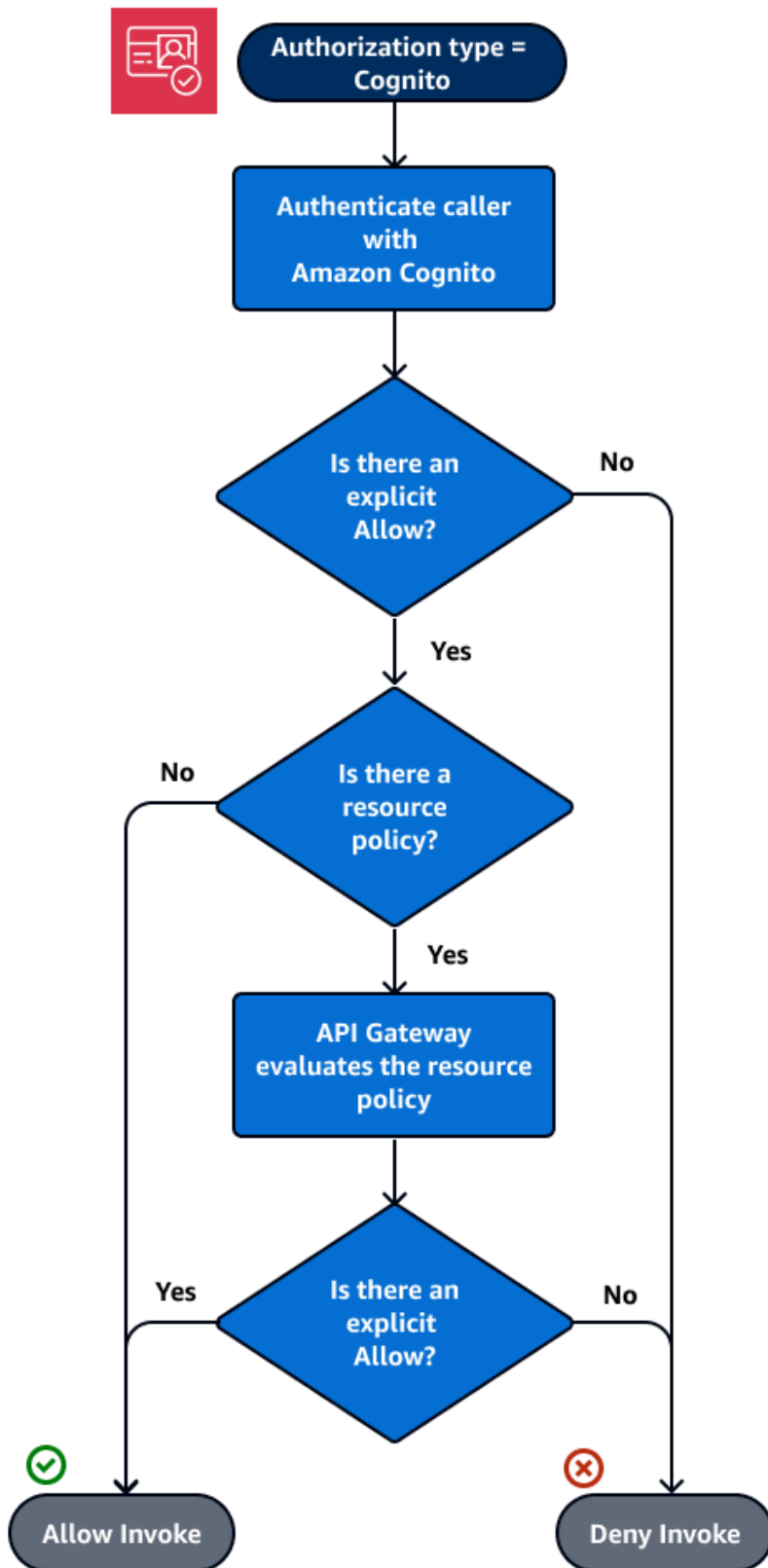
JSON

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

Autenticação e política de recursos do Amazon Cognito

Neste fluxo de trabalho, um [grupo de usuários do Amazon Cognito](#) é configurado para a API, além de uma política de recursos. O API Gateway primeiro tenta autenticar o autor da chamada por meio do Amazon Cognito. Isso é normalmente executado por meio de um [token JWT](#) que é fornecido pelo autor da chamada. Se a autenticação for bem-sucedida, a política de recursos é avaliada de forma independente e é necessária uma permissão explícita. Uma negação ou “nem permissão nem negação” resulta em uma negação. Veja a seguir um exemplo de uma política de recursos que pode ser usada com os grupos de usuários do Amazon Cognito.



Veja a seguir um exemplo de uma política de recursos que permite chamadas somente de IPs de origem especificados, pressupondo que o token de autenticação do Amazon Cognito contém uma permissão. Consulte mais informações em [Tabela 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"]
        }
      }
    }
  ]
}
```

Tabelas de resultados de avaliação de política

A Tabela A lista o comportamento resultante quando o acesso a uma API do API Gateway é controlado por uma política do IAM ou um autorizador do Lambda e por uma política de recursos do API Gateway, ambos na mesma Conta da AWS.

Política do IAM (ou autorizador do Lambda)	Política de recursos do API Gateway	Comportamento resultante
Permitir	Permitir	Permitir
Permitir	Nem permitir ou negar	Permitir
Permitir	Deny	Negação explícita
Nem permitir ou negar	Permitir	Permitir

Política do IAM (ou autorizador do Lambda)	Política de recursos do API Gateway	Comportamento resultante
Nem permitir ou negar	Nem permitir ou negar	Negação implícita
Nem permitir ou negar	Deny	Negação explícita
Deny	Permitir	Negação explícita
Deny	Nem permitir ou negar	Negação explícita
Deny	Deny	Negação explícita

A Tabela B lista o comportamento resultante quando o acesso a uma API do API Gateway é controlado por uma política do IAM ou um autorizador de grupos de usuários do Amazon Cognito e por uma política de recursos do API Gateway, os quais estão em Contas da AWS diferentes. Se uma delas for silenciosa (nem permissão nem negação), o acesso entre contas é negado. Isso ocorre porque o acesso entre contas requer que tanto a política de recursos quanto a política do IAM, ou o autorizador de grupos de usuários do Amazon Cognito, conceda acesso explicitamente.

Política do IAM (ou autorizador de grupos de usuários do Amazon Cognito)	Política de recursos do API Gateway	Comportamento resultante
Permitir	Permitir	Permitir
Permitir	Nem permitir ou negar	Negação implícita
Permitir	Deny	Negação explícita
Nem permitir ou negar	Permitir	Negação implícita
Nem permitir ou negar	Nem permitir ou negar	Negação implícita
Nem permitir ou negar	Deny	Negação explícita
Deny	Permitir	Negação explícita
Deny	Nem permitir ou negar	Negação explícita

Política do IAM (ou autorizador de grupos de usuários do Amazon Cognito)	Política de recursos do API Gateway	Comportamento resultante
Deny	Deny	Negação explícita

Exemplos de política de recursos do API Gateway

Esta página apresenta alguns exemplos de casos de uso típicos de políticas de recursos do API Gateway.

Os exemplos de política a seguir usam uma sintaxe simplificada para especificar o recurso da API. Essa sintaxe simplificada é uma forma abreviada de como você pode se referir a um recurso de API, em vez de especificar o nome de recurso da Amazon (ARN) completo. O API Gateway converte a sintaxe abreviada para o ARN completo quando você salva a política. Por exemplo, você pode especificar o recurso `execute-api:/stage-name/GET/pets` em uma política de recursos. O API Gateway converte o recurso para `arn:aws:execute-api:us-east-2:123456789012:aabbccdde/stage-name/GET/pets` quando você salva a política de recursos. O API Gateway cria o ARN completo usando a região atual, o ID da conta da AWS e o ID da API REST à qual a política de recursos está associada. É possível usar `execute-api:/*` para representar todos os estágios, métodos e caminhos na API atual. Para obter informações sobre a linguagem de políticas de acesso, consulte [Visão geral da linguagem de políticas de acesso para o Amazon API Gateway](#).

Tópicos

- [Exemplo: permitir que funções em outra conta da AWS usem uma API](#)
- [Exemplo: negar tráfego da API com base no intervalo ou endereço IP de origem](#)
- [Exemplo: negar tráfego de API com base no endereço IP ou intervalo de origem ao usar uma API privada](#)
- [Exemplo: permitir tráfego da API privada com base na VPC ou no VPC endpoint de origem](#)

Exemplo: permitir que funções em outra conta da AWS usem uma API

O exemplo de política de recursos a seguir concede acesso à API em uma conta da AWS a dois perfis em outra conta da AWS por meio dos protocolos [Signature Version 4](#) (SigV4) ou [Signature Version 4a](#) (SigV4a). Especificamente, a função de desenvolvedor e de administrador da conta da

AWS identificada pelo *account-id-2* recebem a ação `execute-api:Invoke` para executar a ação GET no recurso `pets` (API) em sua conta da 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"
      ]
    }
  ]
}
```

Exemplo: negar tráfego da API com base no intervalo ou endereço IP de origem

O exemplo de política de recursos a seguir nega (bloqueia) o tráfego de entrada para uma API de dois blocos de endereços IP de origem especificados.

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" ]
      }
    }
  }
]
}

```

Se você usa políticas de usuário do IAM ou políticas de recursos do API Gateway para controlar o acesso ao API Gateway ou a qualquer API do API Gateway, confirme se suas políticas foram atualizadas para incluir intervalos de endereços IPv6. Políticas que não são atualizadas para lidar com endereços IPv6 podem afetar o acesso do cliente ao API Gateway quando eles começarem a usar o endpoint de pilha dupla. Para obter mais informações, consulte [the section called “Como usar endereços do IPv6 em políticas do IAM”](#).

Exemplo: negar tráfego de API com base no endereço IP ou intervalo de origem ao usar uma API privada

O seguinte exemplo de política de recursos nega (bloqueia) o tráfego de entrada para uma API privada de dois blocos de endereços IP de origem especificados. Ao usar APIs privadas, o VPC endpoint de execute-api regrava o endereço IP de origem. A condição `aws:VpcSourceIp` filtra a solicitação em relação ao endereço IP do 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"]
            }
        }
    }
}
]
}

```

Exemplo: permitir tráfego da API privada com base na VPC ou no VPC endpoint de origem

O exemplo das políticas de recursos a seguir permite o tráfego de entrada para uma API privada apenas proveniente de uma nuvem privada virtual (VPC) ou um VPC endpoint específicos.

Este exemplo de política de recurso especifica uma VPC de origem:

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"
        }
      }
    }
  ]
}

```

Este exemplo de política de recurso especifica um VPC endpoint de origem:

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"
        }
      }
    }
  ]
}

```

```
}  
  }  
] }  
}
```

Criar e anexar uma política de recursos do API Gateway a uma API

Para permitir que um usuário acesse sua API chamando o serviço de execução de API, você precisa criar uma política de recursos do API Gateway e anexar essa política à API. Ao anexar uma política à API, ela aplica as permissões da política aos métodos na API. Se você atualizar a política de recursos, será necessário implantar a API.

Tópicos

- [Pré-requisitos](#)
- [Anexar uma política de recursos a uma API do API Gateway](#)
- [Solução de problemas na política de recursos](#)

Pré-requisitos

Para atualizar uma política de recursos do API Gateway, você precisará ter as permissões `apigateway:UpdateRestApiPolicy` e `apigateway:PATCH`.

Para uma API regional ou otimizada para bordas, é possível anexar a política de recursos à API ao criá-la ou depois de implantá-la. Para uma API privada, não é possível implantar a API sem uma política de recursos. Para obter mais informações, consulte [the section called “APIs REST privadas”](#).

Anexar uma política de recursos a uma API do API Gateway

O procedimento a seguir mostra como anexar uma política de recursos a uma API do API Gateway.

Console de gerenciamento da AWS

Como anexar uma política de recursos a uma API do API Gateway

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST.
3. No painel de navegação principal, escolha Política de recursos.

4. Selecione Create policy (Criar política).
5. (Opcional) Escolha Selecionar um modelo para gerar um exemplo de política.

Nas políticas de exemplo, os espaços reservados são colocados entre chaves duplas ("{{*placeholder*}}"). Substitua cada um dos espaços reservados, incluindo as chaves, pelas informações necessárias.

6. Se você não usar um dos exemplos do modelo, insira a política de recurso.
7. Escolha Salvar alterações.

Se a API tiver sido implantada anteriormente no console do API Gateway, será necessário reimplantá-la para que a política de recursos entre em vigor.

AWS CLI

Para criar uma API por meio da AWS CLI e anexar uma política de recursos a ela, use o comando [create-rest-api](#) indicado abaixo:

```
aws apigateway create-rest-api \  
  --name "api-name" \  
  --policy "{\"jsonEscapedPolicyDocument\"}"
```

Para anexar uma política de recursos a uma API existente por meio da AWS CLI, use o comando [update-rest-api](#) indicado abaixo:

```
aws apigateway update-rest-api \  
  --rest-api-id api-id \  
  --patch-operations op=replace,path=/  
policy,value='\"{\"jsonEscapedPolicyDocument\"}'
```

Você também pode anexar sua política de recursos como um arquivo `policy.json` separado e incluí-la no comando [create-rest-api](#). O comando [create-rest-api](#) indicado abaixo cria uma API com uma política de recurso:

```
aws apigateway create-rest-api \  
  --name "api-name" \  
  --policy file://policy.json
```

`policy.json` é uma política de recursos do API Gateway, como [the section called “Exemplo: negar tráfego da API com base no intervalo ou endereço IP de origem”](#).

AWS CloudFormation

Você pode usar CloudFormation para criar uma API com uma política de recursos. O seguinte exemplo cria uma API REST com o exemplo de política de recursos, [the section called “Exemplo: negar tráfego da API com base no intervalo ou endereço IP de origem”](#).

```
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

```

Solução de problemas na política de recursos

As diretrizes de solução de problemas a seguir podem ajudar a resolver problemas em sua política de recursos.

Minha API retorna {"Message":"Usuário: anônimo não tem autorização para executar: execute-api:Invoke no recurso: arn:aws:execute-api:us-east-1:*****/*/*/*/*"}.

Em sua política de recursos, se você definir a entidade principal como uma entidade principal da AWS, como a seguinte:

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"
    ]
  }
]
}

```

Você deverá usar a autorização AWS_IAM para todos os métodos em sua API, caso contrário, a API retornará a mensagem de erro anterior. Para obter mais instruções sobre como ativar a autorização AWS_IAM para um método, consulte [the section called “Métodos”](#).

Não consigo atualizar minha política de recursos

Se você atualizar a política de recursos depois que a API for criada, será necessário implantar a API para propagar as alterações após a alteração da política. Atualizar ou salvar a política não altera o comportamento do tempo de execução da API. Para mais informações sobre como implantar sua API, consulte [the section called “Implantar APIs REST”](#).

Minha política de recursos retorna o seguinte erro: Documento de política inválido. Verifique a sintaxe da política e se as entidades principais são válidas.

Para solucionar esse erro, primeiro recomendamos que você verifique a sintaxe da política. Para obter mais informações, consulte [the section called “Visão geral da linguagem de políticas de acesso para o Amazon API Gateway”](#). Também recomendamos que você verifique se todas as entidades principais especificadas são válidas e não foram excluídas.

Além disso, se a API estiver em uma [região opcional](#), verifique se todas as contas na política de recursos têm a região habilitada.

AWSAs chaves condições da que podem ser usadas nas políticas de recursos do API Gateway

A tabela a seguir contém as chaves de condição da AWS que podem ser usadas nas políticas de recursos de APIs no API Gateway para cada tipo de autorização.

Para obter mais informações sobre chaves de condição da AWS, consulte [Chaves de contexto de condição global da AWS](#).

Chaves de condição	Critérios	Precisa de AuthN ?	Tipo de autorização
aws:CurrentTime	Nenhum	Não	Tudo

Chaves de condição	Critérios	Precisa de AuthN ?	Tipo de autorização
<code>aws:EpochTime</code>	Nenhum	Não	Tudo
<code>aws:TokenIssueTime</code>	A chave está presente somente em solicitações assinadas com credenciais de segurança temporárias.	Sim	IAM
<code>aws:MultiFactorAuthPresent</code>	A chave está presente somente em solicitações assinadas com credenciais de segurança temporárias.	Sim	IAM
<code>aws:MultiFactorAuthAge</code>	A chave estará presente somente se o MFA estiver presente nas solicitações.	Sim	IAM
<code>aws:PrincipalAccount</code>	Nenhum	Sim	IAM
<code>aws:PrincipalArn</code>	Nenhum	Sim	IAM
<code>aws:PrincipalOrgID</code>	Essa chave será incluída no contexto da solicitação somente se o principal for membro de uma organização.	Sim	IAM

Chaves de condição	Critérios	Precisa de AuthN ?	Tipo de autorização
<code>aws:PrincipalOrgPaths</code>	Essa chave será incluída no contexto da solicitação somente se o principal for membro de uma organização.	Sim	IAM
<code>aws:PrincipalTag</code>	Essa chave será incluída no contexto da solicitação se o principal estiver usando um usuário do IAM com tags anexadas. Ela será incluída para um principal usando uma função do IAM com tags anexadas ou tags de sessão.	Sim	IAM
<code>aws:PrincipalType</code>	Nenhum	Sim	IAM
<code>aws:Referer</code>	A chave estará presente somente se o valor for fornecido pelo autor da chamada no cabeçalho HTTP.	Não	Tudo
<code>aws:SecureTransport</code>	Nenhum	Não	Tudo
<code>aws:SourceArn</code>	Nenhum	Não	Tudo
<code>aws:SourceIp</code>	Nenhum	Não	Tudo

Chaves de condição	Critérios	Precisa de AuthN ?	Tipo de autorização
<code>aws:SourceVpc</code>	Essa chave só pode ser usada para APIs privadas.	Não	Tudo
<code>aws:SourceVpce</code>	Essa chave só pode ser usada para APIs privadas.	Não	Tudo
<code>aws:VpcSourceIp</code>	Essa chave só pode ser usada para APIs privadas.	Não	Tudo
<code>aws:UserAgent</code>	A chave estará presente somente se o valor for fornecido pelo autor da chamada no cabeçalho HTTP.	Não	Tudo
<code>aws:userid</code>	Nenhum	Sim	IAM
<code>aws:username</code>	Nenhum	Sim	IAM

Controlar o acesso a uma API REST com permissões do IAM

O acesso à sua API do Amazon API Gateway com [permissões do IAM](#) é controlado pelo controle do acesso aos dois processos de componente do API Gateway a seguir:

- Para criar, implantar e gerenciar uma API no API Gateway, você deve conceder as permissões de desenvolvedor de APIs para realizar as ações necessárias com suporte pelo componente de gerenciamento de APIs do API Gateway.
- Para chamar uma API implantada ou atualizar o armazenamento em cache de APIs, você deve conceder ao autor da chamada da API as devidas permissões para realizar as ações necessárias do IAM com suporte pelo componente de execução de APIs do API Gateway.

O controle de acesso para os dois processos envolve diferentes modelos de permissões, explicados em seguida.

Modelo de permissões do API Gateway para criar e gerenciar uma API

Para permitir que um desenvolvedor de API crie e gerencie uma API no API Gateway, você deve [criar políticas de permissões do IAM](#) que permitem a um desenvolvedor de API especificado criar, atualizar, implantar, exibir ou excluir [entidades de API](#) necessárias. Você anexa a política de permissões a um usuário, perfil ou grupo.

Para conceder acesso, adicione as permissões aos seus usuários, grupos ou perfis:

- Usuários e grupos no Centro de Identidade do AWS IAM:

Crie um conjunto de permissões. Siga as instruções em [Criação de um conjunto de permissões](#) no Guia do usuário do Centro de Identidade do AWS IAM.

- Usuários gerenciados no IAM com provedor de identidades:

Crie um perfil para a federação de identidades. Siga as instruções em [Criando um perfil para um provedor de identidades de terceiros \(federação\)](#) no Guia do Usuário do IAM.

- Usuários do IAM:

- Crie um perfil que seu usuário possa assumir. Siga as instruções em [Criação de um perfil para um usuário do IAM](#) no Guia do usuário do IAM.
- (Não recomendado) Vincule uma política diretamente a um usuário ou adicione um usuário a um grupo de usuários. Siga as instruções em [Adição de permissões a um usuário \(console\)](#) no Guia do usuário do IAM.

Para obter mais informações sobre como usar esse modelo de permissões, consulte [the section called “Políticas baseadas em identidade do API Gateway”](#).

Modelo de permissões do API Gateway para invocar uma API

Para permitir que um autor da chamada de API invoque a API ou atualize seu armazenamento em cache, é necessário criar políticas do IAM que permitam que um autor da chamada de API específico invoque o método de API para o qual a autenticação de usuários está ativada. O desenvolvedor de APIs define a propriedade `authorizationType` do método como `AWS_IAM` para exigir que o autor da chamada envie as credenciais do usuário do IAM a serem autenticadas. O API Gateway é compatível com o Signature Version 4a (SigV4a) e Signature Version 4 (SigV4) para autenticar as

credenciais do usuário. Para ter mais informações, consulte [AWS Signature Version 4](#). Depois, é possível anexar a política a um usuário, grupo ou perfil.

Nesta instrução de política de permissões do IAM, o elemento `Resource` do IAM contém uma lista de métodos de API implantados identificados por verbos HTTP especificados e por [caminhos de recursos](#) do API Gateway. O elemento `Action` do IAM contém as ações necessárias de execução de API do API Gateway. Essas ações incluem `execute-api:Invoke` ou `execute-api:InvalidCache`, em que `execute-api` designa o componente de execução de API subjacente do API Gateway.

Para obter mais informações sobre como usar esse modelo de permissões, consulte [Controlar o acesso para chamar uma API](#).

Quando uma API é integrada a um serviço da AWS (por exemplo, AWS Lambda) no backend, o API Gateway também deve ter permissões para acessar recursos integrados da AWS (por exemplo, invocar uma função do Lambda) em nome do autor da chamada da API. Para conceder essas permissões, crie uma função do IAM do tipo serviço da AWS para API Gateway. Quando você cria essa função no console de gerenciamento do IAM, essa função resultante contém a seguinte política de confiança do IAM que declara o API Gateway como uma entidade confiável com permissão para assumir a função:

JSON

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

Se você criar a função do IAM chamando o comando [create-role](#) da CLI ou um método de SDK correspondente, forneça a política de confiança acima como o parâmetro de entrada de `assume-`

role-policy-document. Não tente criar essa política diretamente no console de gerenciamento do IAM ou chamando o comando da AWS CLI [create-policy](#) ou um método correspondente do SDK.

Para que o API Gateway chame o serviço da AWS integrado, você também deve anexar a essa função políticas de permissões do IAM apropriadas para chamar os serviços integrados da AWS. Por exemplo, para chamar uma função do Lambda, inclua a seguinte política de permissão do IAM na função do IAM:

JSON

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

Observe que o Lambda oferece suporte a políticas de acesso com base em recursos, que combina políticas de confiança com políticas de permissões. Ao integrar uma API com uma função do Lambda usando o console do API Gateway, você não é solicitado a definir essa função do IAM explicitamente, pois o console define as permissões com base em recurso na função do Lambda para você, com seu consentimento.

Note

Para implementar o controle de acesso em um serviço da AWS, você pode usar o modelo de permissões com base em autor da chamada, no qual uma política de permissões é diretamente anexada ao usuário ou grupo do autor da chamada, ou ao modelo de permissões com base em perfil, no qual uma política de permissões é anexada a um perfil do IAM que o API Gateway pode assumir. As políticas de permissões podem ser diferentes nos dois modelos. Por exemplo, a política baseada em agente de chamada bloqueia o acesso, enquanto a política baseada em função permite o acesso. Você pode utilizar isso para exigir que um usuário acesse um serviço da AWS somente por meio da API do API Gateway.

Controlar o acesso para chamar uma API

Nesta seção, você saberá mais sobre o modelo de permissões para controlar o acesso à sua API usando as permissões do IAM. Quando a autorização do IAM está habilitada, os clientes precisam usar o Signature Version 4a (SigV4a) e o Signature Version 4 (SigV4) para assinar suas solicitações com credenciais da AWS. Para ter mais informações, consulte [AWS Signature Version 4](#).

Nesta seção, mostramos um modelo de declaração de política do IAM e a referência de declarações de política. A referência de declarações de política inclui os formatos dos campos `Action` e `Resource` relacionados ao serviço de execução da API. Use essas referências para criar sua declaração de política do IAM. Ao criar a declaração de política do IAM, talvez seja necessário pensar em como as políticas de recursos do API Gateway afetam o fluxo de trabalho de autorização. Para obter mais informações, consulte [the section called “Como as políticas de recursos afetam o fluxo de trabalho de autorização”](#).

Para APIs privadas, é necessário usar uma combinação de uma política de recursos do API Gateway e de uma política do VPC endpoint. Para obter mais informações, consulte os tópicos a seguir:

- [the section called “Usar políticas de recursos do API Gateway”](#)
- [the section called “Usar políticas de VPC endpoint para APIs privadas”](#)

Controlar quem pode chamar um método de API do API Gateway com políticas do IAM

Para controlar quem pode ou não pode chamar uma API implantada com permissões do IAM, crie um documento de política do IAM com as permissões necessárias. Um modelo para esse documento de política é mostrado da seguinte maneira.

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"
      ]
    }
  ]
}
```

```
    ]
  }
]
}
```

Aqui, é necessário substituir *Permission* por Allow ou Deny dependendo se você deseja conceder ou revogar as permissões incluídas. É necessário substituir *Execution-operation* pelas operações com suporte pelo serviço de execução de API. *METHOD_HTTP_VERB* representa um verbo HTTP com suporte pelos recursos especificados. *Resource-path* é o espaço reservado para o caminho da URL de uma instância de [Resource](#) da API implantada que oferece suporte ao *METHOD_HTTP_VERB* mencionado. Para obter mais informações, consulte [Referência de instrução de políticas do IAM para executar a API no API Gateway](#).

Note

Para que as políticas do IAM sejam eficazes, você deve ter habilitado a autenticação do IAM em métodos de API, definindo `AWS_IAM` para a propriedade `authorizationType` do método. Se isso não for feito, esses métodos de API se tornarão acessíveis ao público.

Por exemplo, para conceder a um usuário a permissão para visualizar uma lista de animais de estimação exposta por uma API especificada, mas negar a esse usuário a permissão para adicionar um animal de estimação à lista, você pode incluir a seguinte instrução na política do 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 um usuário a permissão para visualizar um animal de estimação exposto por uma API que é configurada como GET `/pets/{petId}`, você pode incluir a seguinte instrução na política do 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"
      ]
    }
  ]
}

```

Referência de instrução de políticas do IAM para executar a API no API Gateway

As informações a seguir descrevem o formato de Ação e Recurso das instruções de política do IAM de permissões de acesso para a execução de uma API.

Formato de ação das permissões para executar a API no API Gateway

A expressão `Action` de execução de API possui o seguinte formato geral:

```
execute-api:action
```

em que *action* é uma ação de execução de API disponível:

- `*`, que representa todas as ações a seguir.
- `Invoke`, usado para chamar uma API mediante a solicitação de um cliente.
- `InvalidateCache`, usado para invalidar o cache de API mediante a solicitação de um cliente.

Formato de recurso das permissões para executar a API no API Gateway

A expressão `Resource` de execução de API possui o seguinte formato geral:

```
arn:aws:execute-api:region:account-id:api-id/stage-name/HTTP-VERB/resource-path-specifier
```

em que:

- *region* é a região da AWS (como `us-east-1` ou `*` para todas as regiões da AWS) que corresponde à API implantada para o método.
- *account-id* é o ID de 12 dígitos da conta da AWS do proprietário da API REST.
- *api-id* é o identificador que o API Gateway atribuiu à API para o método.
- *stage-name* é o nome do estágio associado ao método.
- *HTTP-VERB* é o verbo HTTP do método. Pode ser um dos seguintes: GET, POST, PUT, DELETE, PATCH.
- *resource-path-specifier* é o caminho para o método desejado.

Note

Se você especificar um curinga (`*`), a expressão `Resource` aplicará o curinga ao resto da expressão.

Algumas expressões de recursos de exemplo incluem:

- **arn:aws:execute-api:*:*:*** para qualquer caminho de recurso em qualquer estágio, para qualquer API em qualquer região da AWS.
- **arn:aws:execute-api:us-east-1:*:*** para qualquer caminho de recurso em qualquer estágio, para qualquer API na região da AWS us-east-1.
- **arn:aws:execute-api:us-east-1:*:*api-id*/*** para qualquer caminho de recurso em qualquer estágio, para a API com o identificador de *api-id* na região da AWS us-east-1.
- **arn:aws:execute-api:us-east-1:*:*api-id*/test/*** para qualquer caminho de recurso no estágio de test, para a API com o identificador *api-id* na região us-east-1 da AWS.

Para saber mais, consulte [Referência de nome de recurso da Amazon \(ARN\) do API Gateway](#).

Exemplos de políticas do IAM para permissões de execução de API

Para o modelo de permissões e outras informações de segundo plano, consulte [Controlar o acesso para chamar uma API](#).

A instrução de política a seguir concede ao usuário permissão para chamar qualquer método POST ao longo do caminho de mydemoresource, no estágio de test, para a API com o identificador a123456789, supondo que a API correspondente tenha sido implantada na região da 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/*"
      ]
    }
  ]
}
```

O exemplo de instrução de política a seguir concede ao usuário permissão para chamar qualquer método no caminho de recurso de `petstorewalkthrough/pets`, em qualquer estágio, para a API com o identificador de `a123456789`, em qualquer região da AWS em que a API correspondente tenha sido implantada:

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "execute-api:Invoke"
      ],
      "Resource": [
        "arn:aws:execute-api:*:*:a123456789/*/*/petstorewalkthrough/pets"
      ]
    }
  ]
}
```

Usar políticas de VPC endpoint para APIs privadas no API Gateway

Para aumentar a segurança de sua API privada, você pode criar uma política de endpoint da VPC. Uma política de endpoint da VPC é uma política de recursos do IAM que você anexa a um endpoint da VPC. Para obter mais informações, consulte [Controlar o acesso a serviços com endpoints da VPC](#).

Talvez você queira criar uma política de endpoint da VPC para realizar as tarefas a seguir.

- Permitir que somente determinadas organizações ou recursos acessem seu endpoint da VPC e invoquem sua API.
- Usar uma única política e evitar políticas baseadas em sessão ou perfil para controlar o tráfego para sua API.
- Restringir o perímetro de segurança de sua aplicação ao migrar do ambiente on-premises para a AWS.

Política do endpoint da VPC

Veja a seguir as considerações para a política de endpoint da VPC.

- A identidade do invocador é avaliada com base no valor do cabeçalho **Authorization**. A política de endpoint da VPC é avaliada primeiro e, em seguida, o Gateway da API avalia a solicitação com base no tipo de autorização configurada na solicitação do método. A tabela a seguir mostra como a política de endpoint da VPC é avaliada com base no conteúdo do valor do cabeçalho **Authorization**.

Conteúdo do valor do cabeçalho Authorization	Como a política de endpoint da VPC é avaliada
Sem conteúdo	O invocador é avaliado como um usuário anônimo
Assinatura SigV4 ou SigV4a válida	O invocador é avaliado como a identidade IAM autenticada da assinatura
Assinatura SigV4 ou SigV4a inválida	O Gateway da API nega o acesso
Informações de autorização não SigV4, como um token de portador	O invocador é avaliado como um usuário anônimo

- Se seu controle de acesso depender do uso de um token de portador, como um autorizador Lambda ou Amazon Cognito, você pode controlar seu perímetro de segurança usando [propriedades do recurso](#).
- Se seus controles de autorização usarem a autorização do IAM, você poderá controlar seu perímetro de segurança usando [propriedades do recurso](#) e [propriedades da entidade principal](#).
- As políticas de VPC endpoint podem ser usadas com políticas de recurso do API Gateway. A política de recursos do API Gateway é usada para especificar quais entidades principais podem acessar a API. A política de endpoint especifica quem pode acessar a VPC e quais APIs podem ser chamadas pelo endpoint da VPC. Sua API privada precisa de uma política de recursos, mas você não precisa criar uma política de endpoint da VPC personalizada.

Exemplos de política de endpoint da VPC

É possível criar políticas para endpoints da Amazon Virtual Private Cloud para o Amazon API Gateway nas quais você pode especificar o indicado a seguir.

- A entidade principal que pode executar ações.
- As ações que podem ser executadas.
- Os recursos que podem ter ações executadas neles.

Isso pode depender do conteúdo do cabeçalho de autorização. Para obter mais informações, consulte [the section called “Política do endpoint da VPC”](#). Para exemplos adicionais de políticas, consulte [Exemplos de políticas de perímetro de dados](#) no site do GitHub.

Para anexar a política ao VPC endpoint, será necessário usar o console da VPC. Para obter mais informações, consulte [Controlar o acesso a serviços com endpoints da VPC](#).

Exemplo 1: Política de VPC endpoint que concede acesso a duas APIs

O exemplo de política a seguir concede acesso somente a duas APIs específicas por meio do VPC endpoint ao qual a política está anexada.

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/*"
      ]
    }
  ]
}
```

Exemplo 2: Política de VPC endpoint que concede acesso a métodos GET

O exemplo de política a seguir concede aos usuários acesso a métodos GET para uma API específica por meio do VPC endpoint ao qual a política está anexada.

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/*"
      ]
    }
  ]
}
```

Exemplo 3: Política de VPC endpoint que concede acesso a uma API específica para um usuário específico

O exemplo de política a seguir concede acesso a uma API específica para um usuário específico por meio do VPC endpoint ao qual a política está anexada.

Neste caso, como a política restringe o acesso a entidades principais específicas do IAM, você precisa definir o método `authorizationType` como `AWS_IAM` ou `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/*"
    ]
}
]
}

```

Exemplo 4: política de endpoint da VPC que concede aos usuários acesso a um nome de domínio personalizado específico e a todas as APIs associadas ao domínio

O exemplo de política a seguir concede aos usuários acesso a um nome de domínio personalizado específico para APIs privadas por meio do endpoint da VPC ao qual a política está anexada. Com essa política, desde que o usuário tenha criado uma associação de acesso ao nome de domínio entre o endpoint da VPC e o nome de domínio personalizado e tenha permissão para invocar o nome de domínio personalizado e quaisquer APIs privadas associadas ao nome de domínio personalizado, ele poderá invocar quaisquer APIs associadas a esse nome de domínio personalizado. Para obter mais informações, consulte [the section called “Nomes de domínio personalizados para APIs 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",
      }
    }
  }
]
}

```

Exemplo 5: política de endpoint da VPC que concede ou nega acesso a APIs e recursos de domínio específicos

O exemplo de política a seguir concede aos usuários acesso a APIs e recursos de domínio específicos. Com essa política, desde que o usuário tenha criado uma associação de acesso ao nome de domínio entre o endpoint da VPC e o nome de domínio personalizado e tenha permissão para invocar o nome de domínio personalizado e quaisquer APIs privadas associadas ao nome de domínio personalizado, ele poderá invocar APIs privadas e recursos de domínio permitidos. Para obter mais informações, consulte [the section called “Nomes de domínio personalizados para APIs 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/*"
    ]
  }
]
}

```

Exemplo 6: política de endpoint da VPC concedendo ou negando acesso por entidades principais e recursos pertencentes a uma organização

O exemplo de política a seguir concede acesso a entidades principais e recursos que pertencem a uma organização.

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Condition": {
        "StringEquals": {
          "aws:ResourceOrgID": "o-abcd1234",
          "aws:PrincipalOrgID": "o-abcd1234"
        }
      },
      "Action": "*",
      "Resource": "*",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Sid": "AllowRequestsByOrgsIdentitiesToOrgsResources"
    }
  ]
}

```

Usar tags para controlar o acesso a APIs REST no API Gateway

A permissão para acessar APIs REST pode ser ajustada usando o controle de acesso baseado em atributos nas políticas do IAM.

Para obter mais informações, consulte [the section called “Controle de acesso baseado em atributos”](#).

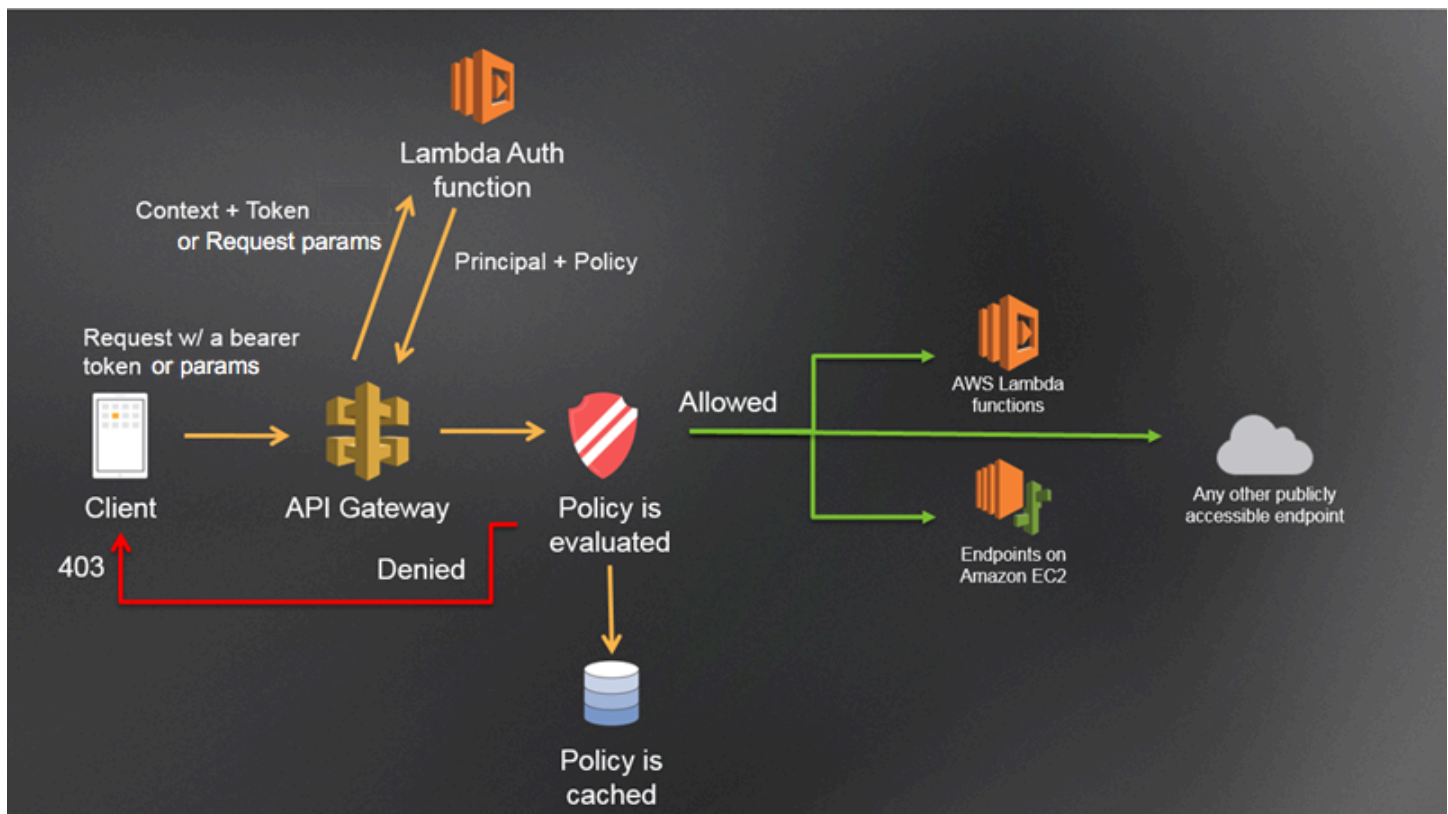
Usar os autorizadores do API Gateway Lambda

Use um autorizador do Lambda (anteriormente conhecido como autorizador personalizado) para controlar o acesso à API. Quando um cliente solicita o método da API, o API Gateway chama o autorizador do Lambda. O autorizador do Lambda usa a identidade do chamador como entrada e retorna uma política do IAM como saída.

Use um autorizador do Lambda para implementar um esquema de autorização personalizado. O esquema pode usar parâmetros de solicitação para determinar a identidade do chamador ou usar uma estratégia de autenticação de token de portador, como OAuth ou SAML. Crie um autorizador do Lambda no console da API REST do API Gateway, usando a AWS CLI ou um SDK da AWS.

Fluxo de trabalho de autorização do autorizador do Lambda

O diagrama a seguir mostra o fluxo de trabalho de autorização referente a um autorizador do Lambda.



Fluxo de trabalho de autorização do Lambda do API Gateway

1. O cliente chama um método em uma API do API Gateway, passando um token de portador ou parâmetros de solicitação.
2. O API Gateway confere se a solicitação do método está configurada com um autorizador do Lambda. Se estiver, o API Gateway chama a função do Lambda.
3. A função do Lambda autentica o chamador. A função pode fazer a autenticação das seguintes maneiras:
 - Chamando um provedor OAuth para receber um token de acesso OAuth.
 - Chamando um provedor SAML para receber uma declaração SAML.
 - Gerando uma política do IAM com base nos valores do parâmetro de solicitação.
 - Recuperando as credenciais de um banco de dados.
4. A função do Lambda retorna uma política do IAM e um identificador de entidade principal. Se a função do Lambda não retornar essas informações, a chamada falhará.
5. O API Gateway avalia a política do IAM.

- Se o acesso for negado, o API Gateway retornará um código de status HTTP, por exemplo, 403 ACCESS_DENIED.
- Se o acesso for permitido, o API Gateway invocará o método.

Se você habilitar o armazenamento em cache da autorização, o API Gateway armazenará a política em cache para que a função do autorizador do Lambda não seja invocada novamente. Sua política deve ser aplicável a todos os recursos e métodos em sua API.

É possível personalizar as respostas 403 ACCESS_DENIED ou 401 UNAUTHORIZED do gateway. Para saber mais, consulte [the section called “Respostas do gateway”](#).

Como escolher um tipo de autorizador do Lambda

Existem dois tipos de autorizadores do Lambda:

Solicitar um autorizador do Lambda baseado em parâmetros (autorizador **REQUEST**)

Um autorizador REQUEST recebe a identidade do chamador em uma combinação de cabeçalhos, parâmetros de strings de consulta, [stageVariables](#) e variáveis [\\$context](#). É possível usar um autorizador REQUEST para criar políticas refinadas com base nas informações de várias fontes de identidade, como as variáveis de contexto `$context.path` e `$context.httpMethod`.

Se você ativar o armazenamento em cache de autorização para um autorizador REQUEST, o API Gateway verificará se todas as fontes de identidade especificadas estão presentes na solicitação. Se uma fonte de identidade especificada estiver ausente, for nula ou estiver vazia, o API Gateway retornará uma resposta HTTP 401 Unauthorized sem chamar a função do autorizador do Lambda. Quando há várias fontes de identidade definidas, todas são usadas para derivar a chave de cache do autorizador, preservando a ordem. É possível definir uma chave de cache refinada usando várias fontes de identidade.

Se você alterar qualquer parte da chave do cache e reimplantar a API, o autorizador descartará o documento da política em cache e gerará outro.

Se você desativar o armazenamento em cache de autorização para um autorizador REQUEST, o API Gateway passará a solicitação diretamente à função do Lambda.

Autorizador do Lambda com base em token (autorizador **TOKEN**)

Um autorizador TOKEN recebe a identidade do chamador em um token de portador, como um JSON Web Token (JWT) ou um token OAuth.

Se você ativar o armazenamento em cache de autorização para um autorizador TOKEN, o nome do cabeçalho especificado na origem do token será a chave do cache.

Além disso, é possível usar a validação de token para inserir uma instrução RegEx. O API Gateway executa a validação inicial do token de entrada em relação a essa expressão e invoca a função do autorizador do Lambda mediante a validação com êxito. Isso ajuda a reduzir as chamadas para a API.

A propriedade `IdentityValidationExpression` é compatível somente com autorizadores TOKEN. Para obter mais informações, consulte [the section called “x-amazon-apigateway-authorizer”](#).

Note

Recomendamos que você use um autorizador REQUEST para controlar o acesso à API. É possível controlar o acesso à API com base em várias fontes de identidade ao usar um autorizador REQUEST, em comparação com uma única fonte de identidade ao usar um autorizador TOKEN. Além disso, você pode separar as chaves de cache usando várias fontes de identidade para um autorizador REQUEST.

Exemplo de função do Lambda do autorizador **REQUEST**

O código de exemplo a seguir criará uma função do autorizador do Lambda que permite uma solicitação se o cabeçalho `HeaderAuth1`, o parâmetro de consulta `QueryString1` e a variável de estágio de `StageVar1` fornecidos pelo cliente corresponderem aos valores especificados de `headerValue1`, `queryValue1` e `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)
```

Neste exemplo, a função do autorizador do Lambda verifica os parâmetros de entrada e age da seguinte forma:

- Se todos os valores de parâmetro necessários corresponderem aos valores esperados, a função do autorizador retornará uma resposta HTTP 200 OK e uma política do IAM que é semelhante ao seguinte, e a solicitação de método será bem-sucedida:

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/"
    }
  ]
}
```

- Caso contrário, a função do autorizador retornará uma resposta HTTP 401 Unauthorized e ocorrerá uma falha na solicitação do método.

Além de retornar uma política do IAM, a função de autorizador do Lambda também deve retornar o identificador principal do autor da chamada. Opcionalmente, ela pode retornar um objeto `context` com informações adicionais que podem ser passadas ao back-end de integração. Para obter mais informações, consulte [Saída de um autorizador do Lambda para o API Gateway](#).

No código de produção, talvez seja necessário autenticar o usuário antes de conceder a autorização. É possível adicionar a lógica de autenticação à função do Lambda chamando um provedor de autenticação conforme indicado na documentação desse provedor.

Exemplo de função do Lambda do autorizador **TOKEN**

O código de exemplo a seguir cria uma função do Lambda do autorizador TOKEN que permite que um chamador invoque um método caso o valor do token fornecido pelo cliente seja allow. O chamador não tem permissão para invocar a solicitação caso o valor do token seja deny. Se o valor do token for unauthorized ou uma string vazia, a função do autorizador retornará uma resposta 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
```

Neste exemplo, quando a API recebe uma solicitação de método, o API Gateway repassa o token de origem para essa função do autorizador do Lambda no atributo `event.authorizationToken`. A função do autorizador do Lambda lê o token e age da seguinte forma:

- Se o valor do token for `allow`, a função do autorizador retornará uma resposta HTTP 200 OK e uma política do IAM que é semelhante ao seguinte, e a solicitação de método será bem-sucedida:

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/"
    }
  ]
}
```

- Se o valor do token for `deny`, a função do autorizador retornará uma resposta HTTP 200 OK e uma política do IAM Deny que é semelhante ao seguinte, e ocorrerá uma falha na solicitação de 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

Fora do ambiente de teste, o API Gateway retorna uma resposta HTTP 403 Forbidden e ocorre uma falha na solicitação do método.

- Se o valor do token for `unauthorized`, a função do autorizador retornará uma resposta HTTP 401 Unauthorized, e ocorrerá uma falha na chamada de método.
- Se o valor do token for diferente, o cliente receberá uma resposta 500 Invalid token, e ocorrerá uma falha na chamada de método.

Além de retornar uma política do IAM, a função de autorizador do Lambda também deve retornar o identificador principal do autor da chamada. Opcionalmente, ela pode retornar um objeto `context` com informações adicionais que podem ser passadas ao back-end de integração. Para obter mais informações, consulte [Saída de um autorizador do Lambda para o API Gateway](#).

No código de produção, talvez seja necessário autenticar o usuário antes de conceder a autorização. É possível adicionar a lógica de autenticação à função do Lambda chamando um provedor de autenticação conforme indicado na documentação desse provedor.

Exemplos adicionais de funções do autorizador do Lambda

A lista a seguir mostra exemplos adicionais de funções do autorizador do Lambda. É possível criar uma função do Lambda na mesma conta em que você criou a API ou em uma conta diferente.

No exemplo anterior de funções do Lambda, é possível usar a função integrada [AWSLambdaBasicExecutionRole](#), já que essas funções não chamam outros serviços da AWS. Se a função do Lambda chamar outros serviços da AWS, será necessário atribuir um perfil de execução do IAM à função do Lambda. Para criar a função, siga as instruções na [Função de execução AWS Lambda](#).

Exemplos adicionais de funções do autorizador do Lambda

- Para ver um exemplo de aplicação, consulte [Open Banking Brasil: amostras de autorizações](#) no GitHub.
- Para obter mais exemplos de funções do Lambda, consulte [aws-apigateway-lambda-authorizer-blueprints](#) no GitHub.
- É possível criar um autorizador do Lambda que autentica usuários usando grupos de usuários do Amazon Cognito e autoriza chamadores com base em um armazenamento de políticas usando o

Verified Permissions. Para obter mais informações, consulte [the section called “Controlar o acesso com base nos atributos de uma identidade com o Verified Permissions”](#).

- O console do Lambda fornece um esquema Python, que você pode usar ao selecionar Use a blueprint (Usar um esquema) e selecionar o esquema api-gateway-authorizer-python.

Configurar um autorizador do Lambda para o API Gateway

Depois de criar uma função do Lambda, configure-a como um autorizador para a API. Em seguida, configure o método para invocar o autorizador do Lambda e determinar se um chamador pode invocar o método. É possível criar uma função do Lambda na mesma conta em que você criou a API ou em uma conta diferente.

Você pode testar o autorizador do Lambda usando as ferramentas integradas no console do API Gateway ou usando o [Postman](#). Consulte instruções sobre como usar o Postman para testar a função do autorizador do Lambda em [the section called “Chamar uma API com os autorizadores do Lambda”](#).

Configurar um autorizador do Lambda (console)

O procedimento a seguir mostra como criar um autorizador do Lambda no console da API REST do API Gateway. Para saber mais sobre os diferentes tipos de autorizador do Lambda, consulte [the section called “Como escolher um tipo de autorizador do Lambda”](#).

REQUEST authorizer

Como configurar um autorizador **REQUEST** do Lambda

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API e selecione Autorizadores.
3. Selecione Criar autorizador.
4. Em Nome do autorizador, insira um nome para o autorizador.
5. Em Tipo de autorizador, selecione Lambda.
6. Em Função do Lambda, selecione a Região da AWS onde você criou a função do Lambda do autorizador e, depois, insira o nome da função.
7. Mantenha o campo Função Lambda de invocação em branco para permitir que o console da API REST do API Gateway defina uma política baseada em recursos. A política concede permissões ao API Gateway para invocar a função do autorizador do Lambda. Você também

pode optar por inserir o nome de um perfil do IAM para permitir que o API Gateway invoque a função do autorizador do Lambda. Consulte um exemplo de função em [Criar uma função do IAM que pode ser assumida](#).

8. Em Carga de evento do Lambda, escolha Solicitar.
9. Em Tipo de origem de identidade, selecione um tipo de parâmetro. Os tipos de parâmetros com suporte são Header, Query string, Stage variable e Context. Para adicionar mais origens de identidade, escolha Adicionar parâmetro.
10. Para armazenar em cache a política de autorização gerada pelo autorizador, mantenha Armazenamento em cache de autorização ativado. Quando o armazenamento de políticas em cache está habilitado, é possível modificar o valor de TTL. A definição de TTL como zero desabilita o armazenamento em cache de políticas.

Se você habilitar o armazenamento em cache, o autorizador deverá retornar uma política que seja aplicável a todos os métodos em uma API. Para aplicar uma política específica do método, use as variáveis de contexto `$context.path` e `$context.httpMethod`.

11. Selecione Criar autorizador.

TOKEN authorizer

Como configurar um autorizador **TOKEN** do Lambda

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API e selecione Autorizadores.
3. Selecione Criar autorizador.
4. Em Nome do autorizador, insira um nome para o autorizador.
5. Em Tipo de autorizador, selecione Lambda.
6. Em Função do Lambda, selecione a Região da AWS onde você criou a função do Lambda do autorizador e, depois, insira o nome da função.
7. Mantenha o campo Função Lambda de invocação em branco para permitir que o console da API REST do API Gateway defina uma política baseada em recursos. A política concede permissões ao API Gateway para invocar a função do autorizador do Lambda. Você também pode optar por inserir o nome de um perfil do IAM para permitir que o API Gateway invoque a função do autorizador do Lambda. Consulte um exemplo de função em [Criar uma função do IAM que pode ser assumida](#).
8. Em Carga de evento do Lambda, escolha Token.

9. Em Origem do token, insira o nome do cabeçalho que contém o token de autorização. O chamador deve incluir um cabeçalho desse nome para enviar o token de autorização ao autorizador do Lambda.
10. (Opcional) Em Validação do token, insira uma instrução RegEx. O API Gateway executa a validação inicial do token de entrada em relação a esta expressão e invoca o autorizador mediante a validação com êxito.
11. Para armazenar em cache a política de autorização gerada pelo autorizador, mantenha Armazenamento em cache de autorização ativado. Quando o armazenamento em cache de políticas está habilitado, o nome de cabeçalho especificado em Origem do token se torna a chave de cache. Quando o armazenamento de políticas em cache está habilitado, é possível modificar o valor de TTL. A definição de TTL como zero desabilita o armazenamento em cache de políticas.

Se você habilitar o armazenamento em cache, o autorizador deverá retornar uma política que seja aplicável a todos os métodos em uma API. Para aplicar uma política específica do método, desative Armazenamento em cache de autorização.

12. Selecione Criar autorizador.

Depois de criar o autorizador do Lambda, é possível testá-lo. O procedimento a seguir mostra como testar o autorizador do Lambda.

REQUEST authorizer

Como testar um autorizador **REQUEST** do Lambda

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione o nome do autorizador.
3. Em Testar autorizador, insira um valor para a fonte de identidades.

Se você estiver usando o [the section called “Exemplo de função do Lambda do autorizador REQUEST”](#), faça o seguinte:

- a. Selecione Cabeçalho e insira **headerValue1**; depois, escolha Adicionar parâmetro.
- b. Em Tipo de origem de identidade, selecione String de consulta, insira **queryValue1** e escolha Adicionar parâmetro.
- c. Em Tipo de origem de identidade, selecione Variável de estágio e insira **stageValue1**.

Você não pode modificar as variáveis de contexto para a invocação do teste, mas pode modificar o modelo de evento de teste do Autorizador do API Gateway para a função do Lambda. Em seguida, você pode testar a função de autorizador do Lambda com variáveis de contexto modificadas. Para saber mais, consulte [Testing Lambda functions in the console](#) no Guia do desenvolvedor do AWS Lambda.

4. Escolha Testar autorizador.

TOKEN authorizer

Como testar um autorizador **TOKEN** do Lambda

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione o nome do autorizador.
3. Em Testar autorizador, insira um valor para o token.

Se você estiver usando o [the section called “Exemplo de função do Lambda do autorizador TOKEN”](#), faça o seguinte:

- Em authorizationToken, insira **allow**.
4. Escolha Testar autorizador.

Se o autorizador do Lambda negar com sucesso uma solicitação no ambiente de teste, o teste responderá com uma resposta HTTP 200 OK. No entanto, fora do ambiente de teste, o API Gateway retorna uma resposta HTTP 403 Forbidden e ocorre uma falha na solicitação do método.

Configurar um autorizador do Lambda (AWS CLI)

O comando [create-authorizer](#) mostra como criar um autorizador do Lambda usando a AWS CLI.

REQUEST authorizer

O comando [create-authorizer](#) indicado abaixo cria um autorizador REQUEST e usa o cabeçalho Authorizer e a variável de contexto accountId como fontes de identidade:

```
aws apigateway create-authorizer \  
  --rest-api-id 1234123412 \  
  --authorizer-name my-authorizer \  
  --authorizer-type REQUEST \  
  --authorizer-config '{"authorizerUri": "arn:aws:lambda:us-east-1:123456789012:function:my-function"}'
```

```
--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

O comando [create-authorizer](#) indicado abaixo cria um autorizador TOKEN e usa o cabeçalho `Authorization` como fonte de identidade:

```
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
```

Depois de criar o autorizador do Lambda, é possível testá-lo. O comando [test-invoke-authorizer](#) indicado abaixo testa o autorizador do Lambda:

```
aws apigateway test-invoke-authorizer --rest-api-id 1234123412 \  
--authorizer-id efg1234 \  
--headers Authorization='Value'
```

Configurar um método para usar um autorizador do Lambda (console)

Depois de configurar o autorizador do Lambda, anexe-o a um método para a API. Se seu autorizador usa o armazenamento em cache de autorização, atualize a política para controlar o acesso ao método adicional.

Para configurar um método de API para usar um autorizador do Lambda

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione uma API.
3. Selecione Recursos e escolha um novo método ou um já existente.

4. Na guia Solicitação de método, em Configurações de solicitação de método, escolha Editar.
5. Em Autorizador, no menu suspenso, selecione o autorizador do Lambda que você acabou de criar.
6. (Opcional) Se você quiser passar o token de autorização para o back-end, escolha Cabeçalhos de solicitação HTTP. Escolha Adicionar cabeçalho e adicione o nome do cabeçalho de autorização. Em Nome, insira um nome de cabeçalho que corresponda ao nome de Origem do token especificado quando você criou o autorizador do Lambda para a API. Esta etapa não se aplica a autorizadores REQUEST.
7. Escolha Salvar.
8. Escolha Deploy API (Implantar API) para implantar a API em um estágio. Para um autorizador REQUEST que usa variáveis de estágio, você também deve definir as variáveis de estágio necessárias e especificar os respectivos valores enquanto estiver na página Estágio.

Configurar um método para usar um autorizador do Lambda (AWS CLI)

Depois de configurar o autorizador do Lambda, anexe-o a um método para a API. Você pode criar um método ou usar uma operação de patch para anexar um autorizador a um método existente. Se seu autorizador usa o armazenamento em cache de autorização, atualize a política para controlar o acesso ao método adicional.

O comando [put-method](#) indicado abaixo cria um método que usa um autorizador do Lambda:

```
aws apigateway put-method --rest-api-id 1234123412 \  
  --resource-id a1b2c3 \  
  --http-method PUT \  
  --authorization-type CUSTOM \  
  --authorizer-id efg1234
```

O comando [update-method](#) indicado abaixo atualiza um método existente para usar um autorizador do 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 para um autorizador do Lambda do API Gateway

A seção a seguir explica o formato da entrada do API Gateway para um autorizador do Lambda.

Formato de entrada do **TOKEN**

Para um autorizador do Lambda (anteriormente conhecido como autorizador personalizado) do tipo TOKEN, você deve especificar um cabeçalho personalizado como Token Source (Origem do token) quando configurar o autorizador para sua API. O cliente da API deve repassar o token de autorização necessário nesse cabeçalho na solicitação de entrada. Ao receber a solicitação do método de entrada, o API Gateway extrai o token do cabeçalho personalizado. Ele passa o token como a propriedade `authorizationToken` do objeto event da função do Lambda, além do método ARN como a propriedade `methodArn`:

```
{
  "type": "TOKEN",
  "authorizationToken": "{caller-supplied-token}",
  "methodArn": "arn:aws:execute-api:{regionId}:{accountId}:{apiId}/{stage}/{httpVerb}/[{resource}]/[{child-resources}]"
}
```

Neste exemplo, a propriedade `type` especifica o tipo de autorizador, que é um autorizador TOKEN. O `{caller-supplied-token}` origina-se do cabeçalho de autorização em uma solicitação do cliente e pode ser qualquer valor de string. O `methodArn` é o ARN da solicitação de método de entrada e é preenchido pelo API Gateway de acordo com a configuração do autorizador do Lambda.

Formato de entrada da **REQUEST**

Para um autorizador do Lambda do tipo REQUEST, o API Gateway passa os parâmetros de solicitação para a função do Lambda do autorizador como parte do objeto event. Os parâmetros de solicitação incluem cabeçalhos, parâmetros de caminho, parâmetros de string de consulta, variáveis de estágio e algumas das variáveis de contexto de solicitação. O autor da chamada da API pode definir parâmetros de caminho, cabeçalhos e parâmetros de string de consulta. O desenvolvedor da API deve definir as variáveis de estágio durante a implantação da API e o API Gateway fornece o contexto de solicitação em tempo de execução.

Note

Os parâmetros de caminho podem ser passados como parâmetros da solicitação para a função do autorizador do Lambda, mas não podem ser usados como origens de identidade.

O exemplo a seguir mostra uma entrada para um autorizador REQUEST de um método API (GET / request) com uma integração 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"
}
}

```

O `requestContext` é um mapa de pares de chave/valor e corresponde à variável [\\$context](#). Seu resultado é dependente da API.

O API Gateway pode adicionar novas chaves ao mapa. Para obter mais informações sobre a entrada de função do Lambda na integração de proxy do Lambda, consulte [Formato de entrada de uma função do Lambda para integração de proxy](#).

Saída de um autorizador do Lambda para o API Gateway

A saída da função de autorizador do Lambda é um objeto em forma de dicionário, que deve incluir o identificador principal (`principalId`) e um documento de política (`policyDocument`) que contém uma lista de instruções da política. A saída também pode incluir um mapa `context` contendo pares de chave/valor. Se a API usar um plano de uso (o [apiKeySource](#) é definido como `AUTHORIZER`), a função de autorizador do Lambda deve retornar uma das chaves de API do plano de uso como o valor de propriedade `usageIdentifierKey`.

Veja a seguir um exemplo dessa saída.

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}"  
  }  
}
```

Aqui, uma declaração de política especifica se o serviço de execução do API Gateway deve ter permissão ou não (Effect) para invocar (Action) o método de API especificado (Resource). Talvez seja necessário controlar o acesso a vários recursos com base no seu autorizador. Você pode usar um caractere curinga (*) para especificar um tipo de recurso (método). Para obter informações sobre como configurar políticas válidas para chamar uma API, consulte [Referência de instrução de políticas do IAM para executar a API no API Gateway](#).

Para o ARN de um método habilitado para autorização, por exemplo, `arn:aws:execute-api:{regionId}:{accountId}:{apiId}/{stage}/{httpVerb}/{resource}/{child-resources}]`, o tamanho máximo é de 1600 bytes. Os valores de parâmetros de caminho, cujo tamanho é determinado em tempo de execução, podem fazer com que o comprimento do ARN exceda o limite. Quando isso acontecer, o cliente da API receberá uma resposta 414 Request URI too long.

Além disso, o Nome de recurso da Amazon (ARN) do recurso, como mostrado na saída da declaração de política pelo autorizador, atualmente está limitado a 512 caracteres. Por esse motivo, você não deve usar o URI com um token de JWT de um tamanho significativo em um URI de solicitação. Em vez disso, você pode passar o token de JWT com segurança em um cabeçalho de solicitação.

Você pode acessar o valor `principalId` em um modelo de mapeamento usando a variável `$context.authorizer.principalId`. Isso é útil quando você deseja passar o valor para o backend. Para obter mais informações, consulte [Variáveis de contexto para transformações de dados](#).

Você pode acessar o valor `stringKey`, `numberKey` ou `booleanKey` (por exemplo, "value", "1" ou "true") do mapa `context` em um modelo de mapeamento chamando `$context.authorizer.stringKey`, `$context.authorizer.numberKey` ou `$context.authorizer.booleanKey`, respectivamente. Todos os valores retornados são

transformados em string. Observe que não é possível definir um objeto JSON ou matriz como um valor válido de qualquer chave no mapa `context`.

Você pode usar o mapa `context` para retornar credenciais em cache do autorizador para o backend usando um modelo de mapeamento de solicitação de integração. Isso permite que o backend forneça uma experiência de usuário aprimorada usando as credenciais em cache para reduzir a necessidade de acessar as chaves secretas e abrir o tokens de autorização para cada solicitação.

Para a integração de proxy do Lambda, o API Gateway passa o objeto `context` de um autorizador do Lambda diretamente para a função backend do Lambda como parte da entrada `event`.

Você pode recuperar os pares de chave-valor `context` na função do Lambda chamando `$event.requestContext.authorizer.key`.

{api-key} significa uma chave de API no plano de uso do estágio da API. Para obter mais informações, consulte [the section called “Planos de uso”](#).

Veja a seguir a saída de exemplo do autorizador do Lambda de exemplo. A saída de exemplo contém uma declaração de política para bloquear chamadas (Deny) para o método GET no estágio dev de uma API (ymy8tbxw7b) de uma conta da 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/"
      }
    ]
  }
}
```

Chamar uma API com um autorizador do Lambda do API Gateway

Depois de configurar o autorizador do Lambda (anteriormente conhecido como autorizador personalizado) e implantar a API, você deve testar a API com o autorizador do Lambda habilitado. Para isso, você precisa de um cliente REST, como cURL ou [Postman](#). Para os exemplos a seguir, usamos Postman.

Note

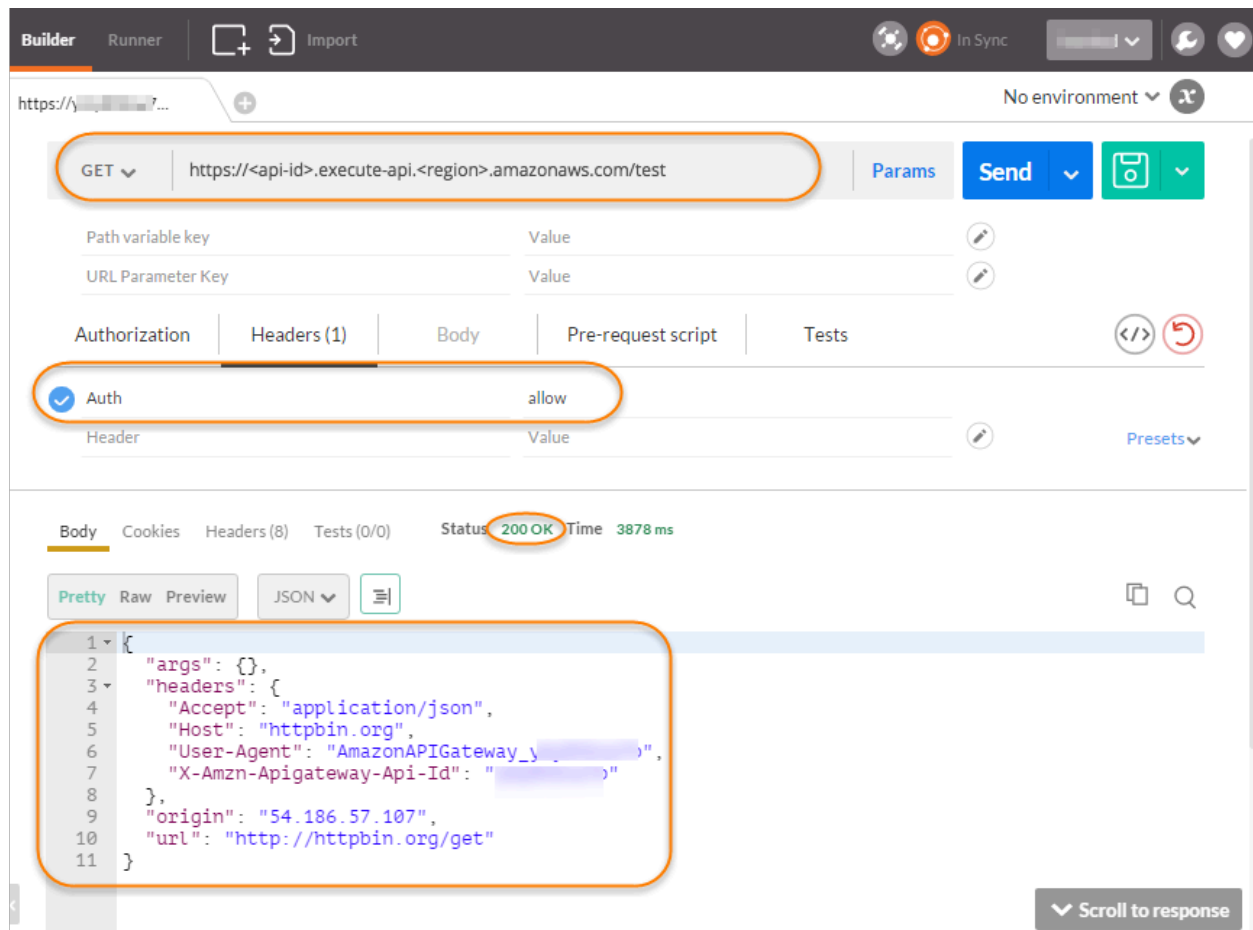
Ao chamar um método habilitado para o autorizador, o API Gateway não registrará a chamada para o CloudWatch se o token necessário para o autorizador TOKEN não estiver definido, for nulo ou for invalidado pela Token validation expression (Expressão de validação do token) especificada. Da mesma forma, o API Gateway não registrará a chamada para o CloudWatch se qualquer uma das origens de identidade necessárias para o autorizador REQUEST não estiver definida ou for nula ou vazia.

No próximo exemplo, mostramos como usar o Postman para chamar ou testar uma API com um autorizador do Lambda TOKEN. O método pode ser aplicado para chamar uma API com um autorizador REQUEST do Lambda, se você especificar explicitamente o caminho, o cabeçalho ou os parâmetros de string de consulta necessários.

Para chamar uma API com o autorizador **TOKEN** personalizado

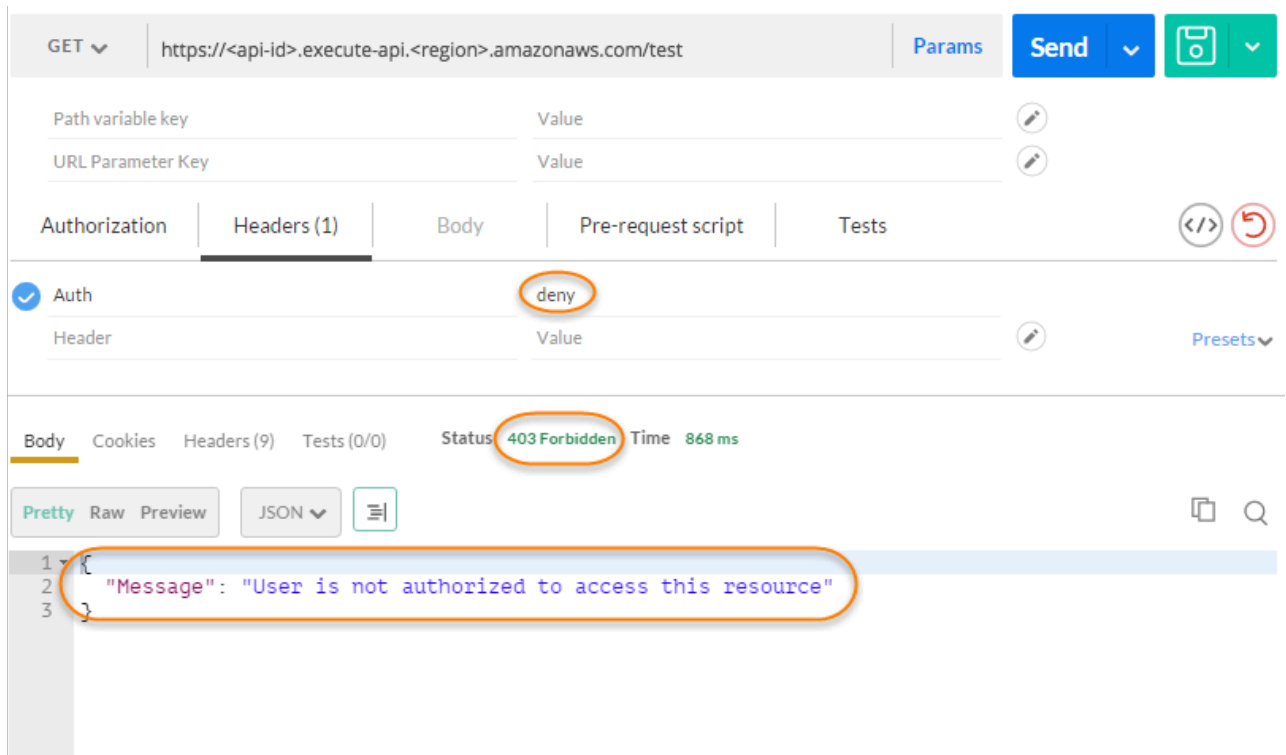
1. Abra Postman, escolha o método GET e cole Invoke URL (Invocar URL) da API no campo de URL adjacente.

Adicione o cabeçalho do token de autorização do Lambda e defina o valor como `allow`.
Selecione Send (Enviar).



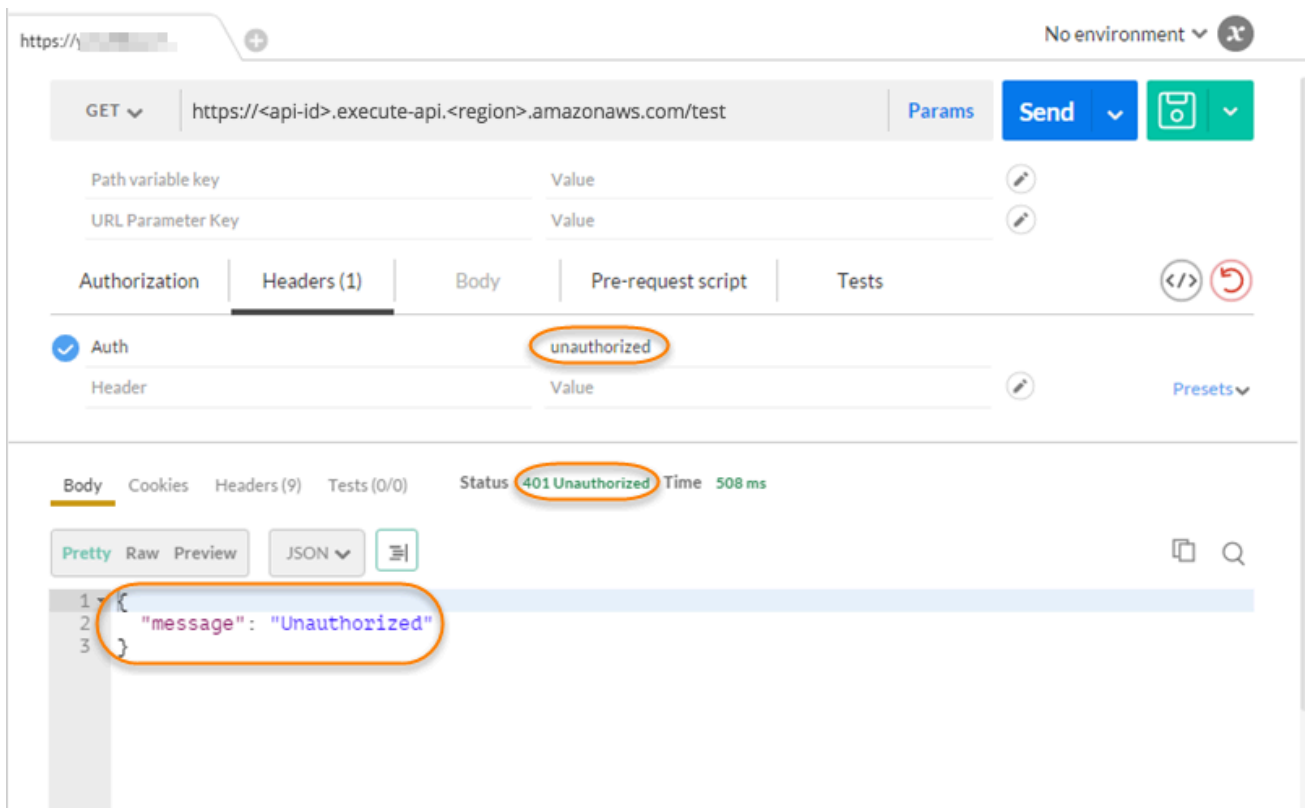
A resposta mostra que o autorizador do Lambda do API Gateway retorna uma resposta 200 OK e autoriza com êxito a chamada para acessar o endpoint HTTP (`http://httpbin.org/get`) integrado com o método.

2. Ainda no Postman, altere o valor do cabeçalho do token de autorização do Lambda para deny. Selecione Send (Enviar).



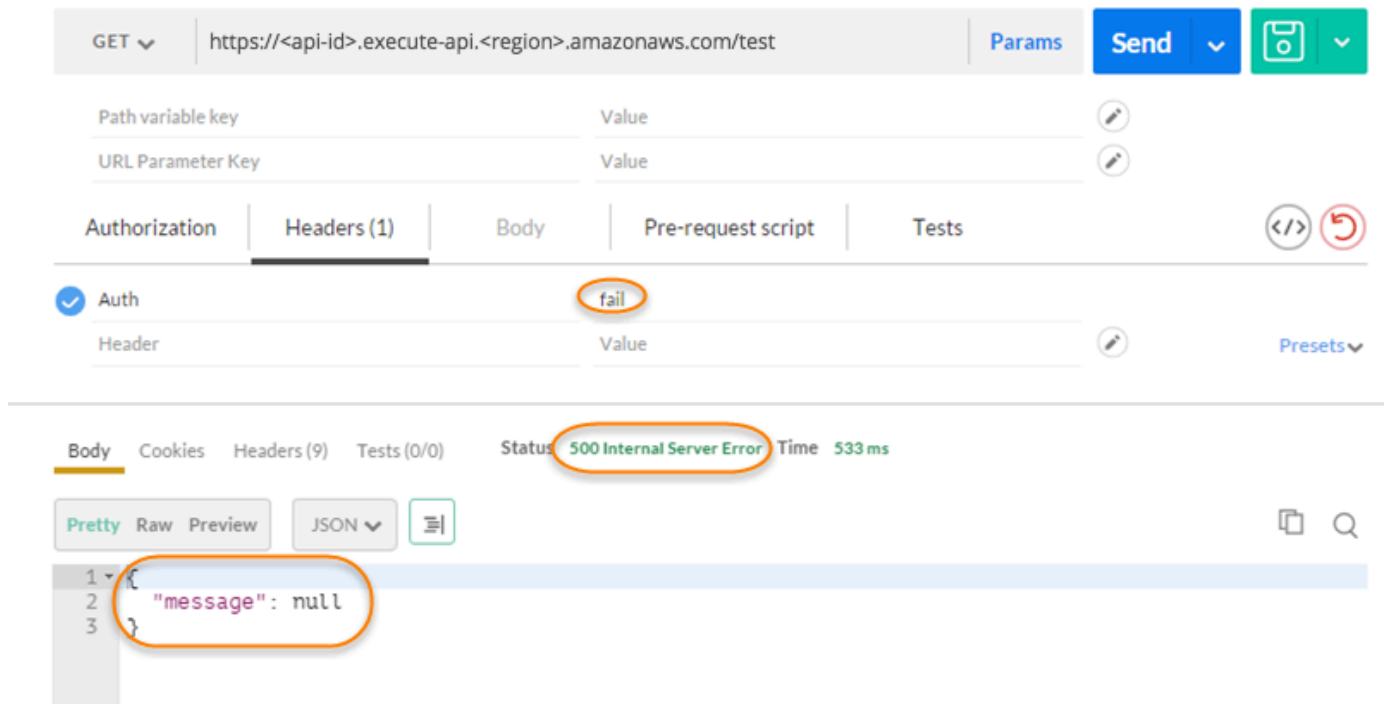
A resposta mostra que o autorizador do Lambda do API Gateway retorna uma resposta 403 Forbidden sem autorizar a chamada para acessar o endpoint HTTP.

3. No Postman, altere o valor do cabeçalho do token de autorização do Lambda para `unauthorized` e escolha Send (Enviar).



A resposta mostra que o API Gateway retorna uma resposta 401 Unauthorized sem autorizar a chamada para acessar o endpoint HTTP.

4. Agora, altere o valor do cabeçalho do token de autorização do Lambda para `fail`. Selecione Send (Enviar).



A resposta mostra que o API Gateway retorna uma resposta 500 Internal Server Error sem autorizar a chamada para acessar o endpoint HTTP.

Configurar um autorizador do Lambda do API Gateway entre contas

Agora você também pode usar uma função do AWS Lambda de uma conta da AWS diferente como sua função do autorizador da API. Cada conta pode estar em qualquer região onde o Amazon API Gateway está disponível. A função do autorizador do Lambda pode usar estratégias de autenticação de token de portador, como OAuth ou SAML. Isso facilita o gerenciamento centralizado e o compartilhamento da função de autorizador do Lambda central em várias APIs do API Gateway.

Nesta seção, mostramos como configurar uma função de autorização do Lambda entre contas usando o console do Amazon API Gateway.

Estas instruções pressupõem que você já tem uma API do API Gateway em uma conta da AWS e uma função do autorizador do Lambda em outra conta.

Como configurar um autorizador do Lambda entre contas usando o console do API Gateway

Faça login no console do Amazon API Gateway na conta que contém a sua API e faça o seguinte:

1. Escolha sua API e, no painel de navegação principal, selecione Autorizadores.

2. Selecione Criar autorizador.
3. Em Nome do autorizador, insira um nome para o autorizador.
4. Em Tipo de autorizador, selecione Lambda.
5. Em Função do Lambda, insira o ARN completo da função do autorizador do Lambda na sua segunda conta.

 Note

No console do Lambda, você pode encontrar o ARN da sua função no canto superior direito da janela.

6. Um aviso com uma string de comando `aws lambda add-permission` será exibido. A política concede permissão do API Gateway para invocar a função do Lambda do autorizador. Copie o comando e salve-o para mais tarde. Execute o comando depois de criar o autorizador.
7. Em Carga útil do evento do Lambda, escolha Token para um autorizador TOKEN ou Solicitação para um autorizador REQUEST.
8. Dependendo da opção da etapa anterior, siga um destes procedimentos:
 - a. Para a opção Token, faça o seguinte:
 - Em Origem do token, insira o nome do cabeçalho que contém o token de autorização. O cliente da API deve incluir um cabeçalho desse nome para enviar o token de autorização ao autorizador do Lambda.
 - Opcionalmente, em Validação do Token, insira uma instrução RegEx. O API Gateway executa a validação inicial do token de entrada em relação a esta expressão e invoca o autorizador mediante a validação com êxito. Isso ajuda a reduzir as chamadas para a API.
 - Para armazenar em cache a política de autorização gerada pelo autorizador, mantenha Armazenamento em cache de autorização ativado. Quando o armazenamento em cache de políticas está habilitado, você pode optar por modificar o valor TTL. A definição de TTL como zero desabilita o armazenamento em cache de políticas. Quando o armazenamento em cache de políticas está habilitado, o nome de cabeçalho especificado em Origem do token se torna a chave de cache. Se vários valores forem passados para esse cabeçalho na solicitação, todos os valores se tornarão a chave de cache, com a ordem preservada.

 Note

O valor de TTL padrão é de 300 segundos. O valor máximo é de 3600 segundos. Não é possível aumentar esse limite.

b. Para a opção de Request (Solicitação), faça o seguinte:

- Em Tipo de origem de identidade, selecione um tipo de parâmetro. Os tipos de parâmetros com suporte são `Header`, `Query string`, `Stage variable` e `Context`. Para adicionar mais origens de identidade, escolha Adicionar parâmetro.
- Para armazenar em cache a política de autorização gerada pelo autorizador, mantenha Armazenamento em cache de autorização ativado. Quando o armazenamento em cache de políticas está habilitado, você pode optar por modificar o valor TTL. A definição de TTL como zero desabilita o armazenamento em cache de políticas.

O API Gateway usa as origens de identidade especificadas como a chave de cache do autorizador de solicitação. Quando o armazenamento em cache é ativado, o API Gateway chama a função do Lambda do autorizador somente depois de verificar com sucesso que todas as origens de identidade especificadas estão presentes em tempo de execução. Se uma origem de identidade especificada estiver ausente, for nula ou vazia, o API Gateway retornará uma resposta 401 `Unauthorized` sem chamar a função do Lambda do autorizador.

Quando várias origens de identidade são definidas, todas são usadas para derivar a chave de cache do autorizador. A alteração de qualquer parte da chave de cache faz com que o autorizador descarte o documento de política armazenado em cache e gere um novo. Se um cabeçalho com vários valores for passado na solicitação, todos os valores farão parte da chave de cache, com a ordem preservada.

- Quando o armazenamento em cache está desativado, não é necessário especificar uma origem de identidade.

 Note

Para habilitar o armazenamento em cache, o autorizador deve retornar uma política que seja aplicável a todos os métodos em uma API. Para aplicar uma política de método específico, você pode desativar Armazenamento em cache de autorização.

9. Selecione Criar autorizador.
10. Cole a string do comando `aws lambda add-permission` que você copiou de uma etapa anterior em uma janela da AWS CLI configurada para sua segunda conta. Substitua `AUTHORIZER_ID` pelo ID do seu autorizador. Isso concederá acesso para sua primeira conta à função do autorizador do Lambda da sua segunda conta.

Controlar o acesso com base nos atributos de uma identidade com o Verified Permissions

Use o Amazon Verified Permissions para controlar o acesso à sua API do API Gateway. Quando você usa o API Gateway com o Verified Permissions, ele cria um autorizador do Lambda que usa decisões de autorização refinadas para controlar o acesso à sua API. O Verified Permissions autoriza os chamadores com base em políticas e em um esquema de armazenamento de políticas usando a linguagem Cedar para definir permissões refinadas para usuários de aplicações. Para saber mais, consulte [Create a policy store with a connected API and identity provider](#) no Guia do usuário do Amazon Verified Permissions.

O Verified Permissions comporta grupos de usuários do Amazon Cognito ou provedores de identidade OpenID Connect (OIDC) como fontes de identidade. O Verified Permissions pressupõe que a entidade principal tenha sido previamente identificada e autenticada. O Verified Permissions só é compatível com APIs REST regionais e otimizadas para borda.

Criar um autorizador do Lambda usando o Verified Permissions

O Verified Permissions cria um autorizador do Lambda para determinar se uma entidade principal pode realizar uma ação em sua API. Você precisa criar a política do Cedar que o Verified Permissions usa para realizar suas tarefas de autorização.

Veja a seguir um exemplo de política do Cedar que permite o acesso para invocar uma API com base no grupo de usuários do Amazon Cognito, `us-east-1_ABC1234`, para o grupo `developer` no recurso `GET /users` de uma API. O Verified Permissions determina a associação ao grupo analisando o token da identidade do chamador.

```
permit(  
  principal in MyAPI::UserGroup::"us-east-1_ABC1234|developer",  
  action in [ MyAPI::Action::"get /users" ],  
  resource  
);
```

O Verified Permissions também pode anexar o autorizador aos métodos da sua API. Nos estágios de produção da sua API, recomendamos não permitir que o Verified Permissions anexe o autorizador para você.

A lista a seguir mostra como configurar o Verified Permissions para anexar ou não o autorizador do Lambda à solicitação dos métodos da sua API.

Anexar o autorizador para você (Console de gerenciamento da AWS)

Ao escolher Criar armazenamento de políticas no console do Verified Permissions, na página Implantar integração de aplicações, escolha Agora.

Não anexe o autorizador para você (Console de gerenciamento da AWS)

Ao escolher Criar armazenamento de políticas no console do Verified Permissions, na página Implantar integração de aplicações, escolha Mais tarde.

O Verified Permissions ainda cria um autorizador do Lambda para você. O autorizador do Lambda começa com `AVPAuthorizerLambda-`. Para ter mais instruções sobre como anexar a autorização em um método, consulte [the section called “Configurar um método para usar um autorizador do Lambda \(console\)”](#).

Anexar o autorizador para você (CloudFormation)

No modelo do CloudFormation gerado pelo Verified Permissions, na seção Conditions, defina `"Ref": "shouldAttachAuthorizer"` como `true`.

Não anexe o autorizador para você (CloudFormation)

No modelo do CloudFormation gerado pelo Verified Permissions, na seção Conditions, defina `"Ref": "shouldAttachAuthorizer"` como `false`.

O Verified Permissions ainda cria um autorizador do Lambda para você. O autorizador do Lambda começa com `AVPAuthorizerLambda-`. Para ter mais instruções sobre como anexar a autorização em um método, consulte [the section called “Configurar um método para usar um autorizador do Lambda \(AWS CLI\)”](#).

Chamar um autorizador do Lambda usando o Verified Permissions

É possível chamar o autorizador do Lambda fornecendo uma identidade ou um token de acesso no cabeçalho `Authorization`. Para obter mais informações, consulte [the section called “Chamar uma API com os autorizadores do Lambda”](#).

O API Gateway armazena em cache a política que o autorizador do Lambda exibe por 120 segundos. É possível modificar o TTL no console do API Gateway ou usando a AWS CLI.

Controlar o acesso a APIs REST usando grupos de usuários do Amazon Cognito como autorizador

Em vez de usar [funções e políticas do IAM](#) ou [autorizadores do Lambda](#) (anteriormente conhecidos como autorizadores personalizados), você pode usar um [grupo de usuários do Amazon Cognito](#) para controlar quem pode acessar sua API no Amazon API Gateway.

Para usar um grupo de usuários do Amazon Cognito com sua API, primeiro é necessário criar um autorizador do tipo COGNITO_USER_POOLS e depois configurar um método da API para usar esse autorizador. Após a implantação da API, o cliente deve primeiro inserir o usuário no grupo de usuários, obter uma [identidade ou token de acesso](#) para o usuário e, então, chamar o método de API com um dos tokens, que são normalmente definidos para o cabeçalho Authorization da solicitação. A chamada de API é realizada somente se o token necessário é fornecido e válido; caso contrário, o cliente não está autorizado a fazer a chamada, pois o cliente não tem credenciais que poderiam ser autorizadas.

O token de identidade é usado para autorizar chamadas de API com base nas declarações de identidade do usuário conectado. O token de acesso é usado para autorizar chamadas de API com base nos escopos personalizados de recursos protegidos por acesso especificado. Para obter mais informações, consulte [Uso de tokens com grupos de usuários](#) e [Servidor de recursos e escopos personalizados](#).

Para criar e configurar um grupo de usuários do Amazon Cognito para sua API, realize as seguintes tarefas:

- Use o console, a CLI/SDK ou a API do Amazon Cognito para criar um grupo de usuários ou use um pertencente a outra conta da AWS.
- Use o console, a CLI/SDK ou a API do API Gateway para criar um autorizador do API Gateway com o grupo de usuários escolhido.
- Use o console, a CLI/SDK ou a API do API Gateway para habilitar o autorizador em métodos de API selecionados.

Para chamar quaisquer métodos de API com um grupo de usuários ativado, os clientes da API realizam as seguintes tarefas:

- Use a CLI/SDK ou a API do Amazon Cognito para conectar um usuário ao grupo de usuários escolhido e obter um token de identidade ou de acesso. Para saber mais sobre o uso dos SDKs, consulte [Exemplos de código do Amazon Cognito usando AWS SDKs](#).
- Use um framework específico do cliente para chamar a API do API Gateway e fornecer o token apropriado no cabeçalho `Authorization`.

Como um desenvolvedor de API, você deve fornecer aos desenvolvedores de clientes o ID do grupo de usuários, um ID de cliente e, possivelmente, os segredos de cliente associado, que são definidos como parte do grupo de usuários.

Note

Para permitir que um usuário faça login usando as credenciais do Amazon Cognito e também obtenha credenciais temporárias para usar com as permissões de uma função do IAM, use as [identidades federadas do Amazon Cognito](#). Para cada método HTTP do endpoint de recurso de API, defina o tipo de autorização, categoria `Method Execution`, como `AWS_IAM`.

Nesta seção, descrevemos como criar um grupo de usuários, integrar uma API do API Gateway a esse grupo de usuários e invocar essa API integrada a ele.

Tópicos

- [Criar um grupo de usuários do Amazon Cognito para uma API REST](#)
- [Integre uma API REST com um grupo de usuários do Amazon Cognito](#)
- [Chamar uma API REST integrada a um grupo de usuários do Amazon Cognito](#)
- [Configurar o autorizador do Amazon Cognito entre contas para uma API REST usando o console do API Gateway](#)
- [Criar um autorizador do Amazon Cognito para uma API REST usando o CloudFormation](#)

Criar um grupo de usuários do Amazon Cognito para uma API REST

Antes de integrar sua API a um grupo de usuários, você deve criar o grupo de usuários no Amazon Cognito. A configuração do grupo de usuários deve seguir todas as [cotas de recursos do Amazon Cognito](#). Todas as variáveis do Amazon Cognito definidas pelo usuário, como grupos, usuários e perfis, devem usar apenas caracteres alfanuméricos. Para obter instruções sobre como criar um

grupo de usuários, consulte [Tutorial: criar um grupo de usuários](#) no Guia do desenvolvedor do Amazon Cognito.

Anote o ID do grupo de usuários, o ID do cliente e qualquer segredo de cliente. O cliente deverá fornecê-los ao Amazon Cognito para que o usuário se registre no grupo de usuários, conecte-se a esse grupo de usuários e obtenha um token de identidade ou acesso a ser incluído em solicitações para chamar métodos de API que são configurados com o grupo de usuários. Além disso, você deve especificar o nome do grupo de usuários ao configurá-lo como um autorizador no API Gateway, conforme descrito a seguir.

Se você estiver usando tokens de acesso para autorizar chamadas de método da API, certifique-se de configurar a integração do aplicativo com o grupo de usuários para definir os escopos personalizados que você deseja em um determinado servidor de recursos. Para obter mais informações sobre o uso de tokens com grupos de usuários do Amazon Cognito, consulte [Uso de tokens com grupos de usuários](#). Para obter mais informações sobre servidores de recursos, consulte [Definir servidores de recursos para seu grupo de usuários](#).

Anote os identificadores do servidor de recursos configurado e os nomes de escopo personalizados. Você precisará deles para elaborar os nomes completos do escopo de acesso para OAuth Scopes (Escopos OAuth), que são usados pelo autorizador do COGNITO_USER_POOLS.

The screenshot shows the Amazon Cognito console interface. The breadcrumb navigation at the top reads: Amazon Cognito > User pools > UserPool6BA7E5F2-H9t6LPh3O3XL > Domain. The left-hand navigation menu includes sections for User management, Authentication, Security, and Branding, with 'Domain' selected under Branding. The main content area is titled 'Domain' and contains two panels. The 'Cognito domain' panel shows 'Domain' as '-' and 'Branding version' as '-'. The 'Custom domain' panel also shows 'Domain' as '-' and 'Branding version' as '-'. Below these is the 'Resource servers (1)' section, which includes a search bar and a table. The table has columns for 'Resource server name', 'Resource server identifier', and 'Custom scopes'. One resource server is listed: 'PetStore' with the identifier 'https://my-petstore-api.example.com' and '2 custom scopes'. The identifier and the custom scopes are highlighted with red boxes in the original image.

Resource server name	Resource server identifier	Custom scopes
PetStore	https://my-petstore-api.example.com	2 custom scopes

Integre uma API REST com um grupo de usuários do Amazon Cognito

Depois de criar um grupo de usuários do Amazon Cognito, no API Gateway, você deve criar um autorizador `COGNITO_USER_POOLS` que use o grupo de usuários. O procedimento a seguir mostra como fazer isso usando o console do API Gateway.

Note

Você pode usar a ação [CreateAuthorizer](#) para criar um autorizador `COGNITO_USER_POOLS` que usa vários grupos de usuários. Você pode usar até 1.000 grupos de usuários para um autorizador `COGNITO_USER_POOLS`. Este limite não pode ser aumentado.

Important

Depois de executar qualquer um dos procedimentos a seguir, você precisará implantar ou reimplantar sua API para propagar as alterações. Para mais informações sobre como implantar sua API, consulte [Implantar APIs REST no API Gateway](#).

Como criar um autorizador **COGNITO_USER_POOLS** usando o console do API Gateway

1. Crie uma nova API ou selecione uma API existente no API Gateway.
2. No painel de navegação principal, selecione Autorizadores.
3. Selecione Criar autorizador.
4. Para configurar o novo autorizador para usar um grupo de usuários, faça o seguinte:
 - a. Em Nome do autorizador, insira um nome.
 - b. Em Tipo de autorizador, selecione Cognito.
 - c. Em Grupo de usuários do Cognito, escolha a Região da AWS onde você criou o Amazon Cognito e selecione um grupo de usuários disponível.

É possível usar uma variável de estágio para definir o grupo de usuários. Use o seguinte formato para o grupo de usuários: `arn:aws:cognito-idp:us-east-2:111122223333:userpool/${stageVariables.MyUserPool}`.

- d. Em Origem do token, insira **Authorization** como o nome de cabeçalho para passar o token de identidade ou acesso que é retornado pelo Amazon Cognito quando um usuário faz login com êxito.
 - e. (Opcional) Insira uma expressão regular no campo Validação do token para validar o campo aud (público) do token de identidade antes de a solicitação ser autorizada com o Amazon Cognito. Observe que, ao usar um token de acesso, essa validação rejeita a solicitação porque o token de acesso não contém o campo aud.
 - f. Selecione Criar autorizador.
5. Depois de criar o autorizador COGNITO_USER_POOLS, você tem a opção de testar a invocação fornecendo um token de identidade provisionado do grupo de usuários. Não é possível usar um token de acesso para testar a invocação do autorizador.

Você pode obter esse token de identidade chamando o [SDK de identidade do Amazon Cognito](#) para fazer login do usuário. Você também pode usar a ação [InitiateAuth](#). Se você não configurar Escopos de autorização, o API Gateway tratará o token fornecido como um token de identidade.

O procedimento anterior cria um autorizador COGNITO_USER_POOLS que usa o grupo de usuários do Amazon Cognito recém-criado. Dependendo de como você habilita o autorizador em um método de API, você pode usar um token de identidade ou de acesso que é provisionado do grupo de usuários integrado.

Para configurar um autorizador do **COGNITO_USER_POOLS** nos métodos

1. Escolha atributos. Selecione um novo método ou escolha um método existente. Se necessário, crie um recurso.
2. Na guia Solicitação de método, em Configurações de solicitação de método, escolha Editar.
3. Para Autorizador, no menu suspenso, selecione os Autorizadores do grupo de usuários do Amazon Cognito que você acabou de criar.
4. Para usar um token de identidade, faça o seguinte:
 - a. Mantenha Escopos de autorização em branco.
 - b. Se necessário, em Solicitação de integração, adicione as expressões `$context.authorizer.claims['property-name']` ou `$context.authorizer.claims.property-name` em um modelo de mapeamento de corpo para transmitir a propriedade de declarações de identidade especificada do

grupo de usuários para o back-end. Para nomes de propriedades simples, como `sub` ou `custom:sub`, as duas notações são idênticas. Para nomes de propriedades complexos, como `custom:role`, você não pode usar a notação de pontos. Por exemplo, as seguintes expressões de mapeamento passam os [campos padrão](#) da declaração de `sub` e `email` para o backend:

```
{
  "context" : {
    "sub" : "$context.authorizer.claims.sub",
    "email" : "$context.authorizer.claims.email"
  }
}
```

Se você tiver declarado um campo de declaração personalizado quando configurou um grupo de usuários, poderá seguir o mesmo padrão para acessar os campos personalizados. O exemplo a seguir obtém um campo `role` personalizado de uma declaração:

```
{
  "context" : {
    "role" : "$context.authorizer.claims.role"
  }
}
```

Se o campo de declaração personalizada for declarado como `custom:role`, use o exemplo a seguir para obter a propriedade da declaração:

```
{
  "context" : {
    "role" : "$context.authorizer.claims['custom:role']"
  }
}
```

5. Para usar um token de acesso, faça o seguinte:

- a. Em Escopos de autorização, insira um ou mais nomes completos de um escopo que tenha sido configurado quando o grupo de usuários do Amazon Cognito foi criado. Por exemplo, seguindo o exemplo apresentado em [Criar um grupo de usuários do Amazon Cognito para uma API REST](#), um dos escopos é `https://my-petstore-api.example.com/cats.read`.

Durante o runtime, a chamada do método ocorre com êxito se qualquer escopo especificado no método desta etapa corresponder a um escopo que foi reivindicado no token de entrada. Caso contrário, a chamada falhará com uma resposta 401 `Unauthorized`.

b. Escolha Salvar.

6. Repita essas etapas para outros métodos que você quiser.

Com o autorizador `COGNITO_USER_POOLS`, se a opção `OAuth Scopes` não estiver especificada, o API Gateway trata o token fornecido como um token de identidade e verifica a identidade declarada em relação à do grupo de usuários. Caso contrário, o API Gateway trata o token fornecido como um token de acesso e verifica os escopos de acesso que são declarados no token em relação aos escopos de autorização declarados no método.

Em vez de usar o console do API Gateway, você também pode habilitar um grupo de usuários do Amazon Cognito em um método especificando um arquivo de definição do OpenAPI e importando a definição da API para o API Gateway.

Para importar um autorizador `COGNITO_USER_POOLS` com um arquivo de definição do OpenAPI

1. Crie (ou exporte) um arquivo de definição do OpenAPI para a sua API.
2. Especifique a definição JSON do autorizador `COGNITO_USER_POOLS` (`MyUserPool`) como parte da seção `securitySchemes` no OpenAPI 3.0 ou da seção `securityDefinitions` no OpenAPI 2.0 da seguinte forma:

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 usar o token de identidade para a autorização de método, adicione { "MyUserPool": [] } à definição security do método, conforme mostrado no seguinte método GET no recurso raiz.

```

"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. Para usar o token de acesso para a autorização do método, mude a definição de segurança acima para { "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. Se necessário, é possível definir outras configurações de API usando as extensões ou definições do OpenAPI apropriadas. Para obter mais informações, consulte [Extensões OpenAPI para o API Gateway](#).

Chamar uma API REST integrada a um grupo de usuários do Amazon Cognito

Para chamar um método com um autorizador de grupo de usuários configurado, o cliente deve fazer o seguinte:

- Permitir que o usuário se inscreva no grupo de usuários.
- Permitir que o usuário se conecte ao grupo de usuários.
- Obtenha uma [token de acesso ou identidade](#) do usuário conectado no grupo de usuários.
- Inclua o token no cabeçalho Authorization (ou outro cabeçalho especificado quando você criou o autorizador).

Você pode usar o [AWS Amplify](#) para executar essas tarefas. Para obter mais informações, consulte [Como integrar o Amazon Cognito com aplicações Web e móveis](#).

- Para Android, consulte [Conceitos básicos do Amplify para Android](#).
- Para usar o iOS, consulte [Conceitos básicos do Amplify para iOS](#).
- Para usar o JavaScript, consulte [Conceitos básicos do Amplify para Javascript](#).

Configurar o autorizador do Amazon Cognito entre contas para uma API REST usando o console do API Gateway

Agora você também pode usar um grupo de usuários do Amazon Cognito de uma conta da AWS diferente como seu autorizador de API. O grupo de usuários do Amazon Cognito pode usar estratégias de autenticação de token de portador, como OAuth ou SAML. Isso facilita o gerenciamento central e o compartilhamento de um autorizador central do grupo de usuários do Amazon Cognito em várias APIs do API Gateway.

Nesta seção, mostramos como configurar um grupo de usuários do Amazon Cognito entre contas usando o console do Amazon API Gateway.

Essas instruções pressupõem que você já tem uma API do API Gateway em uma conta da AWS e um grupo de usuários do Amazon Cognito em outra conta.

Configurar o autorizador do Amazon Cognito entre contas para uma API REST

Faça login no console do Amazon API Gateway na conta que contém a sua API e faça o seguinte:

1. Crie uma nova API ou selecione uma API existente no API Gateway.
2. No painel de navegação principal, selecione Autorizadores.
3. Selecione Criar autorizador.
4. Para configurar o novo autorizador para usar um grupo de usuários, faça o seguinte:
 - a. Em Nome do autorizador, insira um nome.
 - b. Em Tipo de autorizador, selecione Cognito.
 - c. Em Grupo de usuários do Cognito, copie e cole o ARN completo do grupo de usuários de sua segunda conta.

 Note

No console do Amazon Cognito é possível encontrar o ARN do grupo de usuários no campo Pool ARN (ARN do grupo) do painel General Settings (Configurações gerais).

- d. Em Origem do token, insira **Authorization** como o nome de cabeçalho para passar o token de identidade ou acesso que é retornado pelo Amazon Cognito quando um usuário faz login com êxito.
- e. (Opcional) Insira uma expressão regular no campo Validação do token para validar o campo aud (público) do token de identidade antes de a solicitação ser autorizada com o Amazon Cognito. Observe que, ao usar um token de acesso, essa validação rejeita a solicitação porque o token de acesso não contém o campo aud.
- f. Selecione Criar autorizador.

Criar um autorizador do Amazon Cognito para uma API REST usando o CloudFormation

É possível usar o CloudFormation para criar um grupo de usuários e um autorizador do Amazon Cognito. O modelo do CloudFormation de exemplo faz o seguinte:

- Criar um grupo de usuários do Amazon Cognito. O cliente deve primeiro inscrever o usuário no grupo de usuários e obter uma [identidade ou um token de acesso](#). Se você estiver usando tokens de acesso para autorizar chamadas de método da API, certifique-se de configurar a integração do aplicativo com o grupo de usuários para definir os escopos personalizados que você deseja em um determinado servidor de recursos.
- Cria uma API do API Gateway com um método GET.
- Cria um autorizador do Amazon Cognito que usa o cabeçalho Authorization como fonte do 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
            - /

```

Integrações para APIs REST no API Gateway

Após a configuração de um método de API, você deve integrá-la a um endpoint no backend. Um endpoint de backend também é denominado endpoint de integração e pode ser uma função do Lambda, uma página da web HTTP ou uma ação de serviço da AWS.

Assim como ocorre com o método de API, a integração de API tem uma solicitação e uma resposta de integração. Uma solicitação de integração encapsula uma solicitação HTTP recebida pelo backend. Ela pode ou não ser diferente da solicitação de método enviada pelo cliente. A resposta de integração é uma resposta HTTP que encapsula a saída retornada pelo backend.

A configuração de uma solicitação de integração envolve o seguinte: configurar como transmitir solicitações de método enviadas pelo cliente para o backend; configurar como transformar os dados da solicitação, se necessário, para os dados da solicitação de integração; especificar qual função do Lambda chamar, especificar para qual servidor HTTP encaminhar a solicitação recebida ou especificar a ação de serviço da AWS a ser chamada.

A configuração de uma resposta de integração (aplicável somente a integrações não proxy), envolve o seguinte: configurar como transmitir o resultado retornado pelo backend para uma resposta de método de um determinado código de status, configurar como transformar os parâmetros de resposta de integração especificados em parâmetros de resposta de método pré-configurados e configurar como mapear o corpo de resposta de integração para o corpo de resposta de método de acordo com os modelos de mapeamento de corpo especificados.

Programaticamente, uma solicitação de integração é encapsulada pelo recurso [Integration](#) e uma resposta de integração pelo recurso [IntegrationResponse](#) do API Gateway.

Para configurar uma solicitação de integração, crie um recurso [Integration](#) e use-o para configurar a URL de endpoint de integração. Depois, defina as permissões do IAM para acessar o backend e especifique os mapeamentos para transformar os dados da solicitação recebida antes de transmiti-los para o backend. Para configurar uma resposta para integração não proxy, crie um recurso [IntegrationResponse](#) e use-o para definir seu método de resposta de destino. Em seguida, configure como mapear a saída de backend para a resposta de método.

Tópicos

- [Configurar uma solicitação de integração no API Gateway](#)
- [Configurar uma resposta de integração no API Gateway](#)
- [Integrações do Lambda para APIs REST no API Gateway](#)
- [Integrações de HTTP para APIs REST no API Gateway](#)
- [Transmitir a resposta para suas integrações de proxy no API Gateway](#)
- [Integrações privadas para APIs REST no API Gateway](#)
- [Simular integrações para APIs REST no API Gateway](#)

Configurar uma solicitação de integração no API Gateway

Para configurar uma solicitação de integração, execute as seguintes tarefas obrigatórias e opcionais:

1. Escolha um tipo de integração que determine como os dados da solicitação de método são passados para o backend.
2. Para integrações não simuladas, especifique um método HTTP e o URI do endpoint de destino da integração, exceto para a integração MOCK.
3. Para integrações com funções do Lambda e outras ações de serviço da AWS, defina uma função do IAM com as permissões necessárias para o API Gateway chamar o backend em seu nome.
4. Para integrações não proxy, defina os mapeamentos de parâmetros necessários para mapear os parâmetros de solicitação de método predefinidos para os parâmetros de solicitação de integração apropriados.
5. Para integrações não proxy, defina os mapeamentos de corpo necessários para mapear o corpo de solicitação de método de entrada de um determinado tipo de conteúdo de acordo com o modelo de mapeamento especificado.
6. Para integrações não proxy, especifique a condição na qual os dados de solicitação de método de entrada são transmitidos para o backend no estado em que se encontram.
7. Opcionalmente, especifique como lidar com a conversão de tipo para uma carga útil binária.
8. Opcionalmente, declare um nome de namespace de cache e os parâmetros de chave de cache para habilitar o armazenamento em cache da API.

A execução dessas tarefas envolve a criação de um recurso [Integração](#) do API Gateway e a definição dos valores de propriedade apropriados. Você pode fazer isso usando o console do API Gateway, os comandos da AWS CLI, um AWS SDK ou a API REST do API Gateway.

Tópicos

- [Tarefas básicas de uma solicitação de integração da API](#)
- [Escolher um tipo de integração de API do API Gateway](#)
- [Configurar a integração de proxy com um recurso de proxy](#)
- [Configurar uma solicitação de integração de API usando o console do API Gateway](#)

Tarefas básicas de uma solicitação de integração da API

Uma solicitação de integração é uma solicitação HTTP que o API Gateway envia para o backend, transmitindo os dados da solicitação enviada pelo cliente e transformação dos dados, se necessário. O método HTTP (ou o verbo) e o URI da solicitação de integração são determinados pelo backend (ou seja, o endpoint de integração). Eles podem ser os mesmos ou diferentes do método HTTP da solicitação de método e do URI, respectivamente.

Por exemplo, quando uma função do Lambda retorna um arquivo que é procurado no Amazon S3, você pode expor essa operação intuitivamente como uma solicitação de método GET para o cliente, embora a solicitação de integração correspondente exija que uma solicitação POST seja usada para invocar a função do Lambda. Para um endpoint HTTP, é provável que a solicitação de método e a solicitação de integração correspondente usem o mesmo verbo HTTP. No entanto, isso não é obrigatório. Você pode integrar a seguinte solicitação de método:

```
GET /{var}?query=value
Host: api.domain.net
```

Com a solicitação de integração a seguir:

```
POST /
Host: service.domain.com
Content-Type: application/json
Content-Length: ...

{
  path: "{var}'s value",
  type: "value"
}
```

Como desenvolvedor de API, você pode usar qualquer verbo HTTP e URI para que uma solicitação de método atenda aos seus requisitos. No entanto, você deve seguir os requisitos do endpoint de integração. Quando os dados da solicitação de método diferem dos dados da solicitação de integração, você pode reconciliar a diferença ao fornecer mapeamentos dos dados da solicitação de método para os dados da solicitação de integração.

Nos exemplos anteriores, o mapeamento converte a variável de caminho ({var}) e os valores do parâmetro de consulta (query) da solicitação de método GET nos valores das propriedades da carga útil da solicitação de integração de path e type. Outros dados de solicitação mapeáveis incluem

cabeçalhos e corpo da solicitação. Eles estão descritos em [the section called “Mapeamento de parâmetros”](#).

Ao configurar a solicitação de integração de proxy HTTP ou o HTTP, você atribui a URL de endpoint HTTP de backend como o valor de URI da solicitação de integração. Por exemplo, na API PetStore, a solicitação, o método para obter uma página de animais de estimação tem o seguinte URI de solicitação de integração:

```
http://petstore-demo-endpoint.execute-api.com/petstore/pets
```

Ao configurar o Lambda ou a integração do proxy do Lambda, você atribui o Nome de recurso da Amazon (ARN) para invocar a função do Lambda como o valor da solicitação de integração do URI. Esse Nome de recurso da Amazon (ARN) tem o seguinte formato:

```
arn:aws:apigateway:api-region:lambda:path//2015-03-31/functions/arn:aws:lambda:lambda-region:account-id:function:lambda-function-name/invocations
```

A parte após `arn:aws:apigateway:api-region:lambda:path/`, ou seja, `/2015-03-31/functions/arn:aws:lambda:lambda-region:account-id:function:lambda-function-name/invocations`, é o caminho do URI da API REST da ação [Invocar](#) do Lambda. Se você usar o console do API Gateway para configurar a integração do Lambda, o API Gateway cria o ARN e o atribui ao URI da integração depois de solicitar a escolha de *lambda-function-name* a partir de uma região.

Depois de configurar a solicitação de integração com outra ação de serviço da AWS, o URI da solicitação de integração também é um ARN, semelhante à integração com a ação `Invoke` do Lambda. Por exemplo, para a integração com a ação [GetBucket](#) do Amazon S3, o URI da solicitação de integração é um ARN do seguinte formato:

```
arn:aws:apigateway:api-region:s3:path/{bucket}
```

O URI da solicitação de integração segue a convenção de caminho para especificar a ação, em que *{bucket}* é o espaço reservado de um nome de bucket. Como alternativa, uma ação de serviço da AWS pode ser referenciada por seu nome. Usando o nome da ação, o URI da solicitação de integração para a ação `GetBucket` do Amazon S3 se torna o seguinte:

```
arn:aws:apigateway:api-region:s3:action/GetBucket
```

Com a URI da solicitação de integração com base em ação, o nome do bucket (`{bucket}`) deve ser especificado no corpo da solicitação de integração (`{ Bucket: "{bucket}" }`), seguindo o formato de entrada da ação `GetBucket`.

Para integrações da AWS, você também deve configurar [credenciais](#) para permitir que o API Gateway chame as ações integradas. Você pode criar uma função nova do IAM ou escolher uma existente para o API Gateway chamar a ação e especificar a função usando seu ARN. Veja a seguir um exemplo desse Nome de recurso da Amazon (ARN):

```
arn:aws:iam::account-id:role/iam-role-name
```

Essa função do IAM deve conter uma política para permitir que a ação seja executada. Ela também deve ter o API Gateway declarado (na relação de confiança da função) como uma entidade confiável para assumir a função. Essas permissões podem ser concedidas na própria ação. Elas são conhecidas como permissões com base em recursos. Para a integração do Lambda, você pode chamar a ação [addPermission](#) do Lambda para definir as permissões baseadas em recursos e definir `credentials` como nulas na solicitação de integração do API Gateway.

Já abordamos a configuração de integração básica. As configurações avançadas envolvem dados da solicitação de método de mapeamento para os dados da solicitação de integração. Para obter mais informações, consulte [the section called “Transformações de dados”](#).

Escolher um tipo de integração de API do API Gateway

Escolha o tipo de integração da API de acordo com os tipos de parâmetro de integração usados e com o modo desejado para transmitir dados que têm origem ou destino no endpoint de integração. Para uma função do Lambda, você pode ter a integração do proxy do Lambda ou a integração personalizada do Lambda. Para um endpoint HTTP, você pode ter a integração de proxy HTTP ou a integração personalizada HTTP. Para uma ação de serviço da AWS, você só pode ter a integração da AWS do tipo não proxy. O API Gateway também oferece suporte à integração simulada, na qual o API Gateway serve como um endpoint de integração para responder a uma solicitação de método.

A integração personalizada do Lambda é um caso especial da integração da AWS em que o endpoint de integração corresponde à ação [function-invoking do serviço](#) do Lambda.

Programaticamente, você escolhe um tipo de integração definindo a propriedade [type](#) no recurso [Integration](#). Para a integração de proxy do Lambda, o valor é `AWS_PROXY`. Para a integração personalizada do Lambda e todas as outras integrações da AWS, é `AWS`. Para a integração de

proxy HTTP e integração HTTP, o valor é HTTP_PROXY e HTTP, respectivamente. Para a integração simulada, o valor `type` é MOCK.

A integração proxy do Lambda oferece suporte a uma configuração de integração simplificada com uma única função do Lambda. A configuração é simples e pode evoluir com o backend sem a necessidade de descartar a configuração existente. Por esses motivos, é altamente recomendável para a integração com uma função do Lambda.

Por outro lado, a integração personalizada do Lambda permite a reutilização de modelos de mapeamento configurados para vários endpoints de integração que tenham requisitos semelhantes dos formatos de dados de entrada e saída. A configuração é mais complexa e é recomendável para cenários de aplicativos mais avançados.

Da mesma forma, a integração de proxy HTTP tem uma configuração de integração simplificada e pode evoluir com o backend sem a necessidade de descartar a configuração existente. A integração personalizada HTTP é mais difícil de configurar, mas permite a reutilização de modelos de mapeamento configurados para outros endpoints de integração.

A lista a seguir resume os tipos de integração suportados:

- **AWS:** esse tipo de integração permite que uma API exponha ações de serviço da AWS. Na integração AWS, você deve configurar a solicitação e a resposta de integração e configurar os mapeamentos de dados necessários da solicitação de método para a solicitação de integração, e da resposta de integração para a resposta de método.
- **AWS_PROXY:** esse tipo de integração permite que um método de API seja integrado à invocação da função do Lambda com uma configuração de integração flexível, versátil e simplificada. Essa integração se baseia em interações diretas entre o cliente e a função integrada do Lambda.

Com esse tipo de integração, também conhecida como integração de proxy do Lambda, você não define a solicitação de integração nem a resposta de integração. O API Gateway transmite a solicitação recebida do cliente como a entrada para a função do Lambda do backend. A função integrada do Lambda obtém a entrada [desse formato](#) e analisa a entrada de todas as origens disponíveis, incluindo cabeçalhos de solicitação, variáveis de caminho URL, parâmetros de string de consulta e o corpo aplicável. A função retorna o resultado seguindo esse [formato de saída](#).

Esse é o tipo de integração preferencial para chamar uma função do Lambda por meio do API Gateway e não se aplica a nenhuma outra ação de serviço da AWS, incluindo ações do Lambda diferentes de invocações de funções.

- **HTTP:** Este tipo de integração permite que uma API exponha endpoints HTTP no backend. Com a integração HTTP, também conhecida como integração personalizada HTTP, você deve configurar a solicitação de integração e a resposta de integração. Devem-se configurar os mapeamentos de dados necessários da solicitação de integração com base na solicitação de método e da resposta de método com base na resposta de integração.
- **HTTP_PROXY:** A integração de proxy HTTP permite que um cliente acesse os endpoints HTTP do backend com uma integração simplificada configurada em um único método de API. Você não define a solicitação de integração nem a resposta de integração. O API Gateway transmite a solicitação recebida do cliente para o endpoint HTTP e transmite a resposta de saída do endpoint HTTP para o cliente.
- **MOCK:** Este tipo de integração permite que o API Gateway retorne uma resposta sem enviar a solicitação até o backend. Isso é útil para testes de API, pois ele pode ser usado para testar a configuração da integração sem incorrer em custos pelo uso do backend e para ativar o desenvolvimento colaborativo de uma API.

Em desenvolvimento colaborativo, uma equipe pode isolar o esforço de desenvolvimento configurando simulações de componentes da API pertencentes a outras equipes com o uso das integrações MOCK. Também é usado para retornar cabeçalhos relacionados ao CORS a fim de garantir que o método de API permita acesso ao CORS. Na verdade, o console do API Gateway integra o método OPTIONS para oferecer suporte a CORS com uma integração de simulação.

[Respostas de Gateway](#) são outros exemplos de integrações de simulação.

Configurar a integração de proxy com um recurso de proxy

Para configurar uma integração de proxy em uma API do API Gateway com um [recurso de proxy](#), execute as seguintes tarefas:

- Crie um recurso de proxy com uma variável de caminho voraz de `{proxy+}`.
- Defina o método ANY no recurso de proxy.
- Integre o recurso e o método com um backend usando o tipo de integração HTTP ou Lambda.

Note

Variáveis de caminho vorazes, métodos ANY e tipos de integração de proxy são recursos independentes, embora sejam comumente usados em conjunto. Você pode configurar um

método HTTP específico em um recurso voraz ou pode aplicar tipos de não integração de proxy a um recurso de proxy.

O API Gateway impõe certas restrições e limitações ao lidar com métodos com uma integração de proxy do Lambda ou uma integração de proxy HTTP. Para obter mais detalhes, consulte [the section called “Observações importantes”](#).

 Note

Ao usar a integração de proxy com passagem direta, o API Gateway retornará o cabeçalho `Content-Type:application/json` padrão se o tipo de conteúdo de uma carga útil não for especificado.

Um recurso de proxy é mais poderoso quando integrado a um backend que usa uma integração de proxy HTTP ou uma [integração](#) de proxy do Lambda.

Integração de proxy HTTP com um recurso de proxy

A integração de proxy HTTP, designada por `HTTP_PROXY` na API REST do API Gateway, é para integrar uma solicitação de método com um endpoint HTTP de backend. Com esse tipo de integração, o API Gateway simplesmente passa toda solicitação e resposta entre o front-end e o backend, sujeito a certas [restrições e limitações](#).

 Note

A integração de proxy HTTP oferece suporte para cabeçalhos de vários valores e strings de consulta.

Ao aplicar a integração de proxy HTTP a um recurso de proxy, você pode configurar sua API para expor uma parte ou toda uma hierarquia de endpoint do backend HTTP com uma única configuração de integração. Por exemplo, suponha que o backend do site esteja organizado em várias ramificações de nós de árvore no nó raiz (`/site`) como: `/site/a0/a1/.../aN`, `/site/b0/b1/.../bM` etc. Se você integrar o método ANY no recurso de proxy de `/api/{proxy+}` aos endpoints de backend com caminhos de URL de `/site/{proxy}`, uma única solicitação de integração poderá oferecer suporte a qualquer operação HTTP (GET, POST etc.) em qualquer `[a0, a1, ..., aN, b0, b1, ...bM, ...]`. Se você aplicar uma integração de proxy a um método

HTTP específico, por exemplo, GET, em vez disso, a solicitação de integração resultante funcionará com as operações especificadas (ou seja, GET) em qualquer um desses nós de backend.

Integração de proxy do Lambda com um recurso de proxy

A integração de proxy do Lambda, designada por `AWS_PROXY` na API REST do API Gateway, é para integrar uma solicitação de método com uma função do Lambda no backend. Com esse tipo de integração, o API Gateway aplica um modelo de mapeamento padrão para enviar a solicitação inteira à função do Lambda e transforma a saída da função do Lambda em respostas HTTP.

De maneira semelhante, você pode aplicar a integração de proxy do Lambda a um recurso de proxy de `/api/{proxy+}` para configurar uma única integração para que uma função do Lambda de backend reaja individualmente a alterações em qualquer um dos recursos da API sob `/api`.

Configurar uma solicitação de integração de API usando o console do API Gateway

A configuração de um método de API define o método e descreve seus comportamentos. Para configurar um método, você deve especificar um recurso, incluindo a raiz ("/"), na qual o método está exposto, um método HTTP (GET, POST etc.) e como ele será integrado com o backend de destino. A solicitação e a resposta do método especificam o contrato com o aplicativo iniciador da chamada, estipulando quais parâmetros a API pode receber e qual é a aparência da resposta.

Os procedimentos a seguir descrevem como usar o console do API Gateway para criar uma solicitação de integração.

Tópicos

- [Configurar uma integração com o Lambda](#)
- [Configurar uma integração HTTP](#)
- [Configurar uma integração de serviços da AWS](#)
- [Configurar uma integração simulada](#)

Configurar uma integração com o Lambda

Use uma integração de função do Lambda para integrar a API a uma função do Lambda. No nível da API, esse é um tipo de integração da AWS, se você criar uma integração sem proxy, ou um tipo de integração `AWS_PROXY`, se criar uma integração de proxy.

Configurar uma integração com o Lambda

1. No painel Recursos, escolha Criar método.

2. Em Tipo de método, selecione um método HTTP.
3. Em Integration type (Tipo de integração), escolha Lambda function (Função do Lambda).
4. Para usar uma integração do proxy do Lambda, ative Integração do proxy do Lambda. Para saber mais sobre integrações do proxy do Lambda, consulte [the section called “Compreender a integração de proxy do Lambda”](#).
5. Em Função do Lambda, insira o nome da sua função do Lambda.

Se você estiver usando uma função do Lambda em uma região que não seja a da API, selecione a região no menu suspenso e insira o nome da função do Lambda. Se você estiver usando uma função do Lambda entre contas, insira o ARN da função.

6. Para usar o valor de tempo limite padrão de 29 segundos, deixe a opção Tempo limite padrão ativada. Para definir um tempo limite personalizado, selecione Tempo limite padrão e insira um valor de tempo limite entre 50 e 29000 milissegundos.
7. (Opcional) É possível definir as configurações de solicitação de método usando os menus suspensos a seguir. Escolha Configurações de solicitação de método e defina a solicitação de método. Para saber mais, consulte a etapa 3 de [the section called “Editar uma solicitação de método no console”](#).

Você também pode definir as configurações de solicitação de método depois de criar o método.

8. Escolha Criar método.

Configurar uma integração HTTP

Use uma integração HTTP para integrar a API a um endpoint HTTP. No nível da API, este é o tipo de integração HTTP.

Configurar uma integração HTTP

1. No painel Recursos, escolha Criar método.
2. Em Tipo de método, selecione um método HTTP.
3. Em Tipo de integração, escolha HTTP.
4. Para usar uma integração de proxy HTTP, ative a Integração do proxy HTTP. Para saber mais sobre integrações de proxy HTTP, consulte [the section called “Configurar integrações de proxy HTTP no API Gateway”](#).
5. Em HTTP method (Método HTTP), escolha o tipo de método HTTP mais parecido com o método no backend HTTP.

6. Em URL do endpoint, insira o URL do back-end HTTP que você deseja que esse método use.
7. Em Manuseio de conteúdo, selecione um comportamento de manuseio de conteúdo.
8. Para usar o valor de tempo limite padrão de 29 segundos, deixe a opção Tempo limite padrão ativada. Para definir um tempo limite personalizado, selecione Tempo limite padrão e insira um valor de tempo limite entre 50 e 29000 milissegundos.
9. (Opcional) É possível definir as configurações de solicitação de método usando os menus suspensos a seguir. Escolha Configurações de solicitação de método e defina a solicitação de método. Para saber mais, consulte a etapa 3 de [the section called “Editar uma solicitação de método no console”](#).

Você também pode definir as configurações de solicitação de método depois de criar o método.

10. Escolha Criar método.

Configurar uma integração de serviços da AWS

Use uma integração de serviços da AWS para integrar a API diretamente a um serviço da AWS. No nível da API, este é o tipo de integração AWS.

Para configurar uma API do API Gateway, siga qualquer um destes procedimentos:

- Crie uma nova função do Lambda.
- Defina uma permissão de recurso na função do Lambda.
- Execute qualquer outra ação do serviço do Lambda.

Você deve escolher o Serviço da AWS.

Configurar uma integração de serviços da AWS

1. No painel Recursos, escolha Criar método.
2. Em Tipo de método, selecione um método HTTP.
3. Em Tipo de integração, selecione Serviço da AWS .
4. Para Região da AWS, escolha a região da AWS que esse método deve usar para chamar a ação.
5. Em Serviço da AWS, selecione o serviço da AWS que esse método deve chamar.
6. Em Subdomínio da AWS, digite o subdomínio usado pelo serviço da AWS. Normalmente, você deixaria este campo em branco. Alguns serviços da AWS podem oferecer suporte a

subdomínios como parte dos hosts. Consulte a documentação do serviço para a disponibilidade e, se disponível, detalhes.

7. Para HTTP method (Método HTTP), escolha o tipo de método HTTP que corresponde à ação. Para o tipo de método HTTP, consulte a documentação de referência da API para o serviço da AWS que você escolheu em Serviço da AWS.
8. Em Tipo de ação, selecione Usar nome da ação para usar uma ação de API ou Usar a substituição de caminho para usar um caminho de recurso personalizado. Para conhecer as ações e os caminho de recurso personalizado disponíveis, consulte a documentação de referência de API do serviço da AWS que você escolheu em Serviço da AWS.
9. Insira o Nome da ação ou Substituição do caminho.
10. Em Função de execução, digite o ARN do perfil do IAM que o método usará para chamar a ação.

Para criar um perfil do IAM, é possível adaptar as instruções em [the section called “Etapa 1: Criar o perfil de execução do proxy de serviço da AWS”](#). Especifique uma política de acesso com o número desejado de instruções de recursos e ações: Para obter mais informações, consulte [the section called “Como o Amazon API Gateway funciona com o IAM”](#).

Para a sintaxe da instrução de ação e recurso, consulte a documentação do serviço da AWS que você escolheu em Serviço da AWS.

Para a relação de confiança da função do IAM, especifique o seguinte, que permite que o API Gateway tome medidas em nome da sua conta da AWS:

JSON

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

```
}
```

11. Para usar o valor de tempo limite padrão de 29 segundos, deixe a opção Tempo limite padrão ativada. Para definir um tempo limite personalizado, selecione Tempo limite padrão e insira um valor de tempo limite entre 50 e 29000 milissegundos.
12. (Opcional) É possível definir as configurações de solicitação de método usando os menus suspensos a seguir. Escolha Configurações de solicitação de método e defina a solicitação de método. Para saber mais, consulte a etapa 3 de [the section called “Editar uma solicitação de método no console”](#).

Você também pode definir as configurações de solicitação de método depois de criar o método.

13. Escolha Criar método.

Configurar uma integração simulada

Use uma integração simulada se quiser que o API Gateway atue como seu back-end para retornar respostas estáticas. No nível da API, este é o tipo de integração MOCK. Normalmente, você pode usar a integração MOCK quando a sua API ainda não for final e você quiser gerar respostas de API para desbloquear equipes dependentes para testes. Para o método OPTION, o API Gateway define a integração MOCK como padrão para retornar cabeçalhos com compartilhamento de recursos de origem cruzada para o recurso da API aplicada.

Configurar uma integração simulada

1. No painel Recursos, escolha Criar método.
2. Em Tipo de método, selecione um método HTTP.
3. Em Tipo de integração, escolha Simulação.
4. (Opcional) É possível definir as configurações de solicitação de método usando os menus suspensos a seguir. Escolha Configurações de solicitação de método e defina a solicitação de método. Para saber mais, consulte a etapa 3 de [the section called “Editar uma solicitação de método no console”](#).

Você também pode definir as configurações de solicitação de método depois de criar o método.

5. Escolha Criar método.

Configurar uma resposta de integração no API Gateway

Para uma integração não proxy, é necessário configurar pelo menos uma resposta de integração e transformá-la na resposta padrão, para passar o resultado retornado do backend para o cliente. Você pode optar por transmitir o resultado no estado em que se encontra e transformar os dados de resposta de integração para os dados da resposta de método se os dois tiverem formatos diferentes.

Para uma integração de proxy, o API Gateway transmite automaticamente a saída de backend para o cliente como uma resposta HTTP. Você não define uma resposta de integração nem uma resposta de método.

Para configurar uma resposta de integração, execute as seguintes tarefas obrigatórias e opcionais:

1. Especifique um código de status HTTP de uma resposta de método para a qual os dados da resposta de integração são mapeados. Isso é obrigatório.
2. Defina uma expressão regular para selecionar a saída de backend a ser representada por essa resposta de integração. Se você deixar isso em branco, a resposta será a padrão que é usada para detectar qualquer resposta ainda não configurada.
3. Se necessário, declare mapeamentos consistindo em pares de chave-valor para mapear os parâmetros de resposta de integração especificados para determinados parâmetros de resposta de método.
4. Se necessário, adicione modelos de mapeamento de corpo para transformar cargas úteis de resposta de integração determinadas nas cargas úteis de resposta de método especificadas.
5. Se necessário, especifique como lidar com a conversão de tipo para uma carga útil binária.

Resposta de integração é uma resposta HTTP que encapsula a saída retornada pela resposta do backend. Para um endpoint HTTP, a resposta do backend é uma resposta HTTP. O código de status da resposta de integração pode obter o código de status retornado pelo backend e o corpo da resposta de integração é a carga útil retornada pelo backend. Para obter um endpoint do Lambda, a resposta do backend é a saída retornada do função do Lambda. Com a integração do Lambda a saída da função do Lambda é retornada como uma resposta 200 OK. A carga útil pode conter o resultado como dados JSON, incluindo uma string JSON, um objeto JSON ou uma mensagem de erro como um objeto JSON. Você pode atribuir uma expressão regular à propriedade [selectionPattern](#) para mapear uma resposta de erro para uma resposta de erro HTTP apropriada. Para obter mais informações sobre a resposta do erro da função do Lambda, consulte [Manipular erros do Lambda no API Gateway](#). Com a integração de proxy do Lambda, a função do Lambda deve retornar uma saída com o seguinte 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
}
```

Não há necessidade de mapear a resposta da função do Lambda à sua resposta HTTP adequada.

Para retornar o resultado para o cliente, configure a resposta de integração para transmitir a resposta do endpoint no estado em que se encontra para a resposta de método correspondente. Ou você pode mapear os dados da resposta de método do endpoint aos dados da resposta de método. Os dados da resposta que podem ser mapeados incluem o código de status da resposta, os parâmetros de cabeçalho da resposta e o corpo da resposta. Se nenhuma resposta de método for definida para o código de status retornado, o API Gateway retornará um erro 500. Para obter mais informações, consulte [Substituir parâmetros de solicitação e resposta da API e códigos de status por APIs REST no API Gateway](#).

Integrações do Lambda para APIs REST no API Gateway

Você pode integrar um método de API a uma função do Lambda usando integração de proxy do Lambda ou integração sem proxy do Lambda (personalizada).

Na integração de proxy do Lambda, a configuração necessária é simples. Defina o método HTTP da integração como POST, o URI de endpoint de integração como o ARN da ação de invocação da função do Lambda de uma função específica do Lambda e conceda ao API Gateway permissão para chamar a função do Lambda em seu nome.

Na integração não proxy do Lambda, além das etapas de configuração da integração de proxy, você também especifica como os dados da solicitação de entrada serão mapeados para a solicitação de integração e como os dados da resposta de integração resultante serão mapeados para a resposta de método.

Tópicos

- [Integrações de proxy do Lambda no API Gateway](#)
- [Configurar integrações personalizadas do Lambda no API Gateway](#)
- [Configurar a invocação assíncrona da função do Lambda do backend](#)

- [Manipular erros do Lambda no API Gateway](#)

Integrações de proxy do Lambda no API Gateway

A seção a seguir mostra como usar uma integração de proxy do Lambda.

Tópicos

- [Compreender a integração de proxy do Lambda do API Gateway](#)
- [Suporte para cabeçalhos de vários valores e parâmetros de string de consulta](#)
- [Formato de entrada de uma função do Lambda para integração de proxy](#)
- [Formato de saída de uma função do Lambda para integração de proxy](#)
- [Configurar a integração de proxy do Lambda para o API Gateway usando a AWS CLI](#)
- [Configurar um recurso de proxy com a integração de proxy do Lambda com uma definição OpenAPI](#)

Compreender a integração de proxy do Lambda do API Gateway

A integração de proxy do Lambda do Amazon API Gateway é um mecanismo simples, potente e ágil para criar uma API com uma configuração de um único método de API. A integração de proxy do Lambda permite que o cliente chame uma única função do Lambda no backend. A função acessa muitos recursos ou recursos de outros serviços da AWS, incluindo chamadas para outras funções do Lambda.

Na integração de proxy do Lambda, quando um cliente envia uma solicitação de API, o API Gateway repassa um [objeto do evento](#) para a função do Lambda integrada, exceto pelo fato de que a ordem dos parâmetros da solicitação não é preservada. Esses [dados de solicitação](#) incluem cabeçalhos de solicitação, parâmetros de strings de consulta, variáveis de caminho URL, carga e dados de configuração da API. Os dados de configuração podem incluir o nome do estágio de implantação atual, variáveis de estágio, identidade do usuário ou contexto de autorização (se houver). A função do Lambda do backend analisa os dados da solicitação recebida para determinar a resposta que ela retorna. Para que o API Gateway passe a saída do Lambda como a resposta da API para o cliente, a função do Lambda deve retornar o resultado [neste formato](#).

Como o API Gateway não interfere muito entre o cliente e a função do Lambda do backend para a integração de proxy do Lambda, o cliente e a função do Lambda integrada podem se adaptar às alterações mútuas sem quebrar a configuração de integração existente da API. Para permitir isso, o cliente deve seguir os protocolos de aplicação promulgados pela função do Lambda de backend.

É possível configurar uma integração de proxy do Lambda para qualquer método de API. Mas uma integração de proxy do Lambda é mais potente quando configurada para um método de API que envolve um recurso de proxy genérico. O recurso de proxy genérico pode ser identificado por um modelo variável de caminho especial de `{proxy+}`, pelo espaço reservado de método ANY ou ambos. O cliente pode passar a entrada da função do Lambda de backend na solicitação recebida como parâmetros da solicitação ou carga aplicável. Os parâmetros de solicitação incluem cabeçalhos, variáveis de caminho URL, parâmetros de string de consulta e a carga aplicável. A função do Lambda integrada verifica todas as fontes de entrada antes de processar a solicitação e responder ao cliente com mensagens de erro significativas se qualquer uma das entradas obrigatórias estiver ausente.

Ao chamar um método de API integrado ao método HTTP genérico de ANY e o recurso genérico de `{proxy+}`, o cliente envia uma solicitação com um método HTTP específico em vez de ANY. O cliente também especifica determinado caminho URL em vez de `{proxy+}`, e inclui todos os cabeçalhos necessários, parâmetros de strings de consulta ou uma carga aplicável.

A lista a seguir resume comportamentos de tempo de execução de diferentes métodos de API com a integração de proxy do Lambda:

- ANY `/ {proxy+}`: o cliente deve escolher determinado método HTTP, definir determinada hierarquia de caminho de recursos e pode definir todos os cabeçalhos, parâmetros de strings de consulta e carga aplicáveis para passar os dados como entrada para a função do Lambda integrada.
- ANY `/res`: o cliente deve escolher determinado método HTTP e pode definir todos os cabeçalhos, parâmetros de strings de consulta e carga aplicáveis para passar os dados como entrada para a função do Lambda integrada.
- GET|POST|PUT|... `/ {proxy+}`: o cliente pode definir determinada hierarquia de caminho de recursos, cabeçalhos, parâmetros de strings de consulta e carga aplicáveis para passar os dados como entrada para a função do Lambda integrada.
- GET|POST|PUT|... `/res/{path}/...`: o cliente deve escolher determinado segmento de caminho (para a variável `{path}`) e pode definir os cabeçalhos de solicitação, parâmetros de strings de consulta e carga aplicáveis para passar os dados de entrada para a função do Lambda integrada.
- GET|POST|PUT|... `/res`: o cliente pode escolher os cabeçalhos de solicitação, parâmetros de strings de consulta e carga aplicáveis para passar dados de entrada para a função do Lambda integrada.

Tanto o recurso de proxy de `{proxy+}` quanto o recurso personalizado de `{custom}` são expressos como modelos de variáveis de caminho. No entanto, `{proxy+}` pode indicar qualquer recurso ao longo de uma hierarquia de caminho, enquanto `{custom}` refere-se apenas a determinado segmento de caminho. Por exemplo, um supermercado pode organizar seu inventário de produtos online por nomes de departamento, categorias de produtos e tipos de produtos. O site do supermercado pode representar os produtos disponíveis pelos seguintes modelos de variáveis de caminho de recursos personalizados: `/ {department} / {produce-category} / {product-type}`. Por exemplo, maçãs são representadas por `/produce/fruit/apple` e cenouras por `/produce/vegetables/carrot`. Ele também pode usar `{proxy+}` para representar qualquer departamento, categoria ou tipo de produto que o cliente pode procurar ao fazer compras na loja online. Por exemplo, `{proxy+}` pode indicar qualquer um dos seguintes itens:

- `/produce`
- `/produce/fruit`
- `/produce/vegetables/carrot`

Para permitir que os clientes procurem qualquer produto disponível, a categoria do produto e o departamento associado, você pode expor um único método de GET `/ {proxy+}` com permissões somente leitura. Da mesma forma, para permitir que um supervisor atualize o inventário do departamento do `produce`, você pode configurar outro método único de PUT `/produce/ {proxy+}` com as permissões de leitura/gravação. Para permitir que um caixa atualize o total de um vegetal, você pode configurar um método POST `/produce/vegetables/ {proxy+}` com permissões de leitura/gravação. Para permitir que um gerente de loja realize qualquer ação possível em qualquer produto disponível, o desenvolvedor da loja online pode expor o método ANY `/ {proxy+}` com permissões de leitura/gravação. Em qualquer caso, na ocasião da execução, o cliente ou o funcionário deve selecionar um produto específico de determinado tipo em um departamento escolhido, uma categoria de produto específica em um departamento escolhido ou um departamento específico.

Para obter mais informações sobre como configurar integrações de proxy do API Gateway, consulte [Configurar a integração de proxy com um recurso de proxy](#).

A integração de proxy exige que o cliente tenha um conhecimento mais detalhado dos requisitos de backend. Portanto, para garantir uma melhor performance das aplicações e da experiência do usuário, o desenvolvedor de backend deve comunicar claramente ao desenvolvedor cliente os

requisitos do backend e fornecer um mecanismo de feedback de erro robusto quando os requisitos não são atendidos.

Suporte para cabeçalhos de vários valores e parâmetros de string de consulta

Agora o API Gateway oferece suporte a vários cabeçalhos e parâmetros de string de consulta que possuem o mesmo nome. Cabeçalhos de vários valores, bem como cabeçalhos e parâmetros de valor único, podem ser combinados nas mesmas solicitações e respostas. Para obter mais informações, consulte [Formato de entrada de uma função do Lambda para integração de proxy](#) e [Formato de saída de uma função do Lambda para integração de proxy](#).

Formato de entrada de uma função do Lambda para integração de proxy

Com a integração de proxy do Lambda, o API Gateway mapeia toda a solicitação do cliente para o parâmetro `event` de entrada da função do Lambda de back-end. O exemplo a seguir mostra a estrutura de um evento que o API Gateway envia para uma integração de proxy do Lambda.

Neste exemplo, supomos que a invocação ao API Gateway tenha sido a seguinte:

```
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'
```

A saída será exibida como a seguir:

```
{
  "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

Na entrada:

- A chave `headers` só pode conter cabeçalhos de valor único.
- A chave `multiValueHeaders` pode conter cabeçalhos de vários valores e cabeçalhos de valor único.
- Se você especificar valores para `headers` e `multiValueHeaders`, o API Gateway os mesclará em uma única lista. Se o mesmo de chave/valor for especificado em ambos, somente os valores de `multiValueHeaders` aparecerão na lista mesclada.

Na entrada para a função do Lambda do backend, o objeto `requestContext` é um mapa de pares de chave/valor. Em cada par, a chave é o nome de uma propriedade de variável [\\$context](#), e o valor é o valor dessa propriedade. O API Gateway pode adicionar novas chaves ao mapa.

Dependendo dos recursos que estão habilitados, o mapa `requestContext` pode variar de acordo com a API. Por exemplo, no exemplo anterior, nenhum tipo de autorização é especificado. Portanto, nenhuma propriedade `$context.authorizer.*` ou `$context.identity.*` está presente. Quando um tipo de autorização é especificado, isso faz com que o API Gateway repasse informações do usuário autorizado para o endpoint de integração em um objeto `requestContext.identity` da seguinte forma:

- Quando o tipo de autorização é `AWS_IAM`, as informações do usuário autorizado incluem propriedades `$context.identity.*`.

- Quando o tipo de autorização é COGNITO_USER_POOLS (autorizador do Amazon Cognito), as informações do usuário autorizado incluem propriedades `$context.identity.cognito*` e `$context.authorizer.claims.*`.
- Quando o tipo de autorização é CUSTOM (autorizador do Lambda), as informações do usuário autorizado incluem propriedades `$context.authorizer.principalId` e outras `$context.authorizer.*` aplicáveis.

Formato de saída de uma função do Lambda para integração de proxy

Na integração de proxy do Lambda, o API Gateway requer a função do Lambda do backend para retornar a saída de acordo com o seguinte formato JSON:

```
{
  "isBase64Encoded": true/false,
  "statusCode": httpStatusCode,
  "headers": { "headerName": "headerValue", ... },
  "multiValueHeaders": { "headerName": ["headerValue", "headerValue2", ...], ... },
  "body": "... "
}
```

Na saída:

- As chaves `headers` e `multiValueHeaders` podem ser não especificadas se nenhum cabeçalho de resposta extra precisar ser retornado.
- A chave `headers` só pode conter cabeçalhos de valor único.
- A chave `multiValueHeaders` pode conter cabeçalhos de vários valores e cabeçalhos de valor único. Você pode usar a chave `multiValueHeaders` para especificar todos os seus cabeçalhos extras, incluindo os de valor único.
- Se você especificar valores para `headers` e `multiValueHeaders`, o API Gateway os mesclará em uma única lista. Se o mesmo de chave/valor for especificado em ambos, somente os valores de `multiValueHeaders` aparecerão na lista mesclada.

Para ativar o CORS da integração de proxy do Lambda, você deve adicionar `Access-Control-Allow-Origin:domain-name` à saída `headers`. `domain-name` pode ser `*` para qualquer nome de domínio. A saída `body` é organizado para o front-end como a carga de resposta do método. Se `body` for um blob binário, você poderá codificá-lo como uma string codificada em Base64, definindo

`isBase64Encoded` como `true` e configurando `/*/*` como Binary Media Type (Tipo de mídia binário). Caso contrário, será possível defini-lo como `false` ou deixá-lo sem especificação.

Note

Para obter mais informações sobre como habilitar o suporte a binários, consulte [Ativação do suporte binário usando o console do API Gateway](#). Para obter um exemplo de função do Lambda, consulte [Exibir mídia binária de uma integração do proxy do Lambda no API Gateway](#).

Se a saída da função for de um formato diferente, o API Gateway retorna uma resposta de erro 502 Bad Gateway.

Para retornar uma resposta em uma função do Lambda em Node.js, você poderá usar comandos como o seguinte:

- Para retornar um resultado bem-sucedido, chame `callback(null, {"statusCode": 200, "body": "results"})`.
- Para lançar uma exceção, chame `callback(new Error('internal server error'))`.
- Para um erro no lado do cliente, (se, por exemplo, um parâmetro necessário estiver ausente), você poderá chamar `callback(null, {"statusCode": 400, "body": "Missing parameters of ..."})` para retornar o erro sem lançar uma exceção.

Em uma função do Lambda `async` no Node.js, a sintaxe equivalente seria:

- Para retornar um resultado bem-sucedido, chame `return {"statusCode": 200, "body": "results"}`.
- Para lançar uma exceção, chame `throw new Error("internal server error")`.
- Para um erro no lado do cliente, (se, por exemplo, um parâmetro necessário estiver ausente), você poderá chamar `return {"statusCode": 400, "body": "Missing parameters of ..."} para retornar o erro sem lançar uma exceção.`

Configurar a integração de proxy do Lambda para o API Gateway usando a AWS CLI

Nesta seção, mostramos como usar uma API com a integração de proxy do Lambda usando a AWS CLI. Para obter instruções detalhadas sobre como usar o console do API Gateway para configurar

um recurso de proxy com a integração de proxy do Lambda, consulte [Tutorial: Crie uma API REST com uma integração de proxy do Lambda](#).

Como exemplo, usamos a seguinte função do Lambda como o backend da 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
};
```

Comparando isso à configuração de integração personalizada do Lambda em [the section called “Configurar integrações personalizadas do Lambda”](#), a entrada para essa função do Lambda pode ser expressa nos parâmetros da solicitação e no corpo. Você tem mais latitude para permitir que o cliente transmita os mesmos dados de entrada. Aqui, o cliente pode transmitir o nome do greeter como um parâmetro de string de consulta, um cabeçalho ou uma propriedade de corpo. A função também pode oferecer suporte à integração personalizada do Lambda. A configuração da API é mais simples. Você não configura a resposta de método nem a resposta de integração.

Como configurar uma integração de proxy do Lambda usando a AWS CLI

1. Use o comando [create-rest-api](#) indicado abaixo para criar uma API:

```
aws apigateway create-rest-api --name 'HelloWorld (AWS CLI)'
```

A saída será exibida da seguinte forma:

```
{
  "name": "HelloWorldProxy (AWS CLI)",
  "id": "te6si5ach7",
  "rootResourceId" : "krznpq9xpg",
  "createdDate": 1508461860
}
```

A API id (te6si5ach7) e o rootResourceId (krznpq9xpg) são usados em todo este exemplo.

2. Use o comando [create-resource](#) indicado abaixo para criar um [recurso](#) /greeting do API Gateway:

```
aws apigateway create-resource \
  --rest-api-id te6si5ach7 \
  --parent-id krznpq9xpg \
  --path-part {proxy+}
```

A saída será exibida da seguinte forma:

```
{
  "path": "/{proxy+}",
  "pathPart": "{proxy+}",
  "id": "2jf6xt",
  "parentId": "krznpq9xpg"
}
```

Na próxima etapa, você usa o valor id (2jf6xt) do recurso {proxy+} para criar um método no recurso /{proxy+}.

3. Use o comando [put-method](#) indicado abaixo para criar uma solicitação de método ANY de ANY / {proxy+}:

```
aws apigateway put-method --rest-api-id te6si5ach7 \  
    --resource-id 2jf6xt \  
    --http-method ANY \  
    --authorization-type "NONE"
```

A saída será exibida da seguinte forma:

```
{  
  "apiKeyRequired": false,  
  "httpMethod": "ANY",  
  "authorizationType": "NONE"  
}
```

Esse método de API permite que o cliente receba ou envie saudações da função do Lambda no backend.

4. Use o comando [put-integration](#) indicado abaixo para configurar a integração do método ANY / {proxy+} com uma função do Lambda denominada HelloWorld. Essa função responde à solicitação com uma mensagem de "Hello, {name}!", se o parâmetro greeter for fornecido ou "Hello, World!", se o parâmetro de string de consulta não for definido.

```
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 integrações do Lambda, você deve usar o método HTTP de POST para a solicitação de integração, de acordo com a [especificação da ação do serviço do Lambda para invocações de função](#). A função do IAM de apigAwsProxyRole deve ter políticas que permitem ao serviço apigateway invocar funções do Lambda. Para obter mais

informações sobre as permissões do IAM, consulte [the section called “Modelo de permissões do API Gateway para invocar uma API”](#).

A saída será exibida da seguinte forma:

```
{
  "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"
}
```

Em vez de fornecer um perfil do IAM para `credentials`, você pode usar o comando [add-permission](#) para adicionar permissões baseadas em recurso. Isso é o que o console do API Gateway faz.

5. Use o comando [create-deployment](#) indicado abaixo para implantar a API em um estágio test:

```
aws apigateway create-deployment \
  --rest-api-id te6si5ach7 \
  --stage-name test
```

6. Teste a API usando os seguintes comandos cURL em um terminal.

Chamando a API com o parâmetro de string de consulta de `?greeter=jane`:

```
curl -X GET 'https://te6si5ach7.execute-api.us-west-2.amazonaws.com/test/greeting?greeter=jane'
```

Chamando a API com um parâmetro de cabeçalho 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'
```

Chamando a API com um corpo 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" }'
```

Em todos os casos, a saída é uma resposta 200 com o corpo de resposta a seguir:

```
Hello, jane!
```

Configurar um recurso de proxy com a integração de proxy do Lambda com uma definição OpenAPI

Para configurar um recurso de proxy com o tipo de integração de proxy do Lambda, crie um recurso de API com um parâmetro de caminho voraz (por exemplo, /parent/{proxy+}) e integre esse recurso a um backend da função do Lambda (por exemplo, arn:aws:lambda:us-west-2:123456789012:function:SimpleLambda4ProxyResource) no método ANY. O parâmetro de caminho voraz deve estar no final do caminho do recurso da API. Como no caso de um recurso não proxy, é possível configurar o recurso de proxy usando o console do API Gateway, importando um arquivo de definição do OpenAPI ou chamando diretamente a API REST do API Gateway.

O seguinte arquivo de definição de API do OpenAPI mostra um exemplo de uma API com um recurso de proxy que está integrado à função do Lambda chamada 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"
      }
    }
  }
}
```

```
}  
}
```

Na integração de proxy do Lambda, em tempo de execução, o API Gateway mapeia uma solicitação de entrada no parâmetro `event` de entrada da função do Lambda. A entrada inclui o método de solicitação, o caminho, os cabeçalhos, qualquer parâmetro de consulta, qualquer carga, o contexto associado e quaisquer variáveis de estágio definidas. O formato de entrada é explicado em [Formato de entrada de uma função do Lambda para integração de proxy](#). Para o API Gateway mapear a saída do Lambda para respostas HTTP com êxito, a função do Lambda deve processar a saída do resultado no formato descrito em [Formato de saída de uma função do Lambda para integração de proxy](#).

Com a integração de proxy do Lambda de um recurso de proxy por meio do método ANY, a função do Lambda de backend única serve como o manipulador de eventos para todas as solicitações por meio do recurso de proxy. Por exemplo, para registrar padrões de tráfego, você pode fazer com que um dispositivo móvel envie suas informações de localização de estado, cidade, rua e edifício, enviando uma solicitação com `/state/city/street/house` no caminho da URL para o recurso de proxy. Dessa forma, a função do Lambda de backend pode analisar o caminho da URL e inserir as tuplas de localização em uma tabela do DynamoDB.

Configurar integrações personalizadas do Lambda no API Gateway

Para mostrar como configurar a integração personalizada ou sem proxy do Lambda, criamos uma API do API Gateway para expor o método GET `/greeting?greeter={name}` para invocar uma função do Lambda. Use um dos exemplos de funções do Lambda a seguir para a API.

Use um dos seguintes exemplos de funções do 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

```

A função responde com uma mensagem de "Hello, {name}!" se o valor do parâmetro greeter for uma string não vazia. Ela retornará uma mensagem de "Hello, World!" se o valor greeter for uma string vazia. A função retornará uma mensagem de erro de "Missing the required

greeter parameter." se o parâmetro do greeter não estiver definido na solicitação de entrada. Atribuímos o nome à função HelloWorld.

Ela pode ser criada no console do Lambda ou usando a AWS CLI. Nesta seção, fazemos referência a essa função usando o seguinte Nome de recurso da Amazon (ARN):

```
arn:aws:lambda:us-east-1:123456789012:function:HelloWorld
```

Com a função do Lambda definida no backend, configure a API.

Como configurar a integração personalizada do Lambda usando a AWS CLI

1. Use o comando [create-rest-api](#) indicado abaixo para criar uma API:

```
aws apigateway create-rest-api --name 'HelloWorld (AWS CLI)'
```

A saída será exibida da seguinte forma:

```
{
  "name": "HelloWorld (AWS CLI)",
  "id": "te6si5ach7",
  "rootResourceId" : "krznpq9xpg",
  "createdDate": 1508461860
}
```

A API id (te6si5ach7) e o rootResourceId (krznpq9xpg) são usados em todo este exemplo.

2. Use o comando [create-resource](#) indicado abaixo para criar um [recurso](#) /greeting do API Gateway:

```
aws apigateway create-resource \
  --rest-api-id te6si5ach7 \
  --parent-id krznpq9xpg \
  --path-part greeting
```

A saída será exibida da seguinte forma:

```
{
  "path": "/greeting",
```

```
"pathPart": "greeting",
"id": "2jf6xt",
"parentId": "krznpq9xpg"
}
```

Na próxima etapa, você usa o valor `id` (`2jf6xt`) do recurso `greeting` para criar um método no recurso `/greeting`.

3. Use o comando [put-method](#) indicado abaixo para criar uma solicitação 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
```

A saída será exibida da seguinte forma:

```
{
  "apiKeyRequired": false,
  "httpMethod": "GET",
  "authorizationType": "NONE",
  "requestParameters": {
    "method.request.querystring.greeter": false
  }
}
```

Esse método de API permite que o cliente receba uma saudação da função do Lambda no backend. O parâmetro `greeter` é opcional, pois o backend deve lidar com um autor da chamada anônimo ou um autor da chamada autenticado.

4. Use o comando [put-method-response](#) indicado abaixo para configurar a resposta 200 OK à solicitação 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. Use o comando [put-integration](#) indicado abaixo para configurar a integração do método GET / greeting?greeter={name} com uma função do Lambda denominada HelloWorld. A função responde à solicitação com uma mensagem de "Hello, {name}!", se o parâmetro greeter for fornecido ou "Hello, World!", se o parâmetro de string de consulta não for definido.

```
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
```

O modelo de mapeamento fornecido aqui converte o parâmetro de string de consulta greeter na propriedade greeter da carga útil JSON. Isso é necessário porque a entrada para uma função do Lambda deve ser expressa no corpo.

Important

Para integrações do Lambda, você deve usar o método HTTP de POST para a solicitação de integração, de acordo com a [especificação da ação do serviço do Lambda para invocações de função](#). O parâmetro uri é o Nome de recurso da Amazon (ARN) da ação de invocação da função.

A saída será exibida da seguinte forma:

```
{  
    "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"
}
```

A função do IAM de `apigAwsProxyRole` deve ter políticas que permitem ao serviço `apigateway` invocar funções do Lambda. Em vez de fornecer uma função do IAM para `credentials`, você pode chamar o comando [add-permission](#) para adicionar permissões baseadas em recurso. É assim que o console do API Gateway adiciona essas permissões.

6. Use o comando [put-integration-response](#) indicado abaixo para configurar a resposta de integração e transmitir a saída da função do Lambda ao cliente como a resposta de método 200 OK.

```
aws apigateway put-integration-response \
  --rest-api-id te6si5ach7 \
  --resource-id 2jf6xt \
  --http-method GET \
  --status-code 200 \
  --selection-pattern ""
```

Ao definir o padrão de seleção como uma string vazia, a resposta 200 OK é o padrão.

A saída será exibida da seguinte forma:

```
{
  "selectionPattern": "",
  "statusCode": "200"
}
```

7. Use o comando [create-deployment](#) indicado abaixo para implantar a API em um estágio test:

```
aws apigateway create-deployment \
  --rest-api-id te6si5ach7 \
  --stage-name test
```

8. Teste a API usando o seguinte comando cURL em um 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 a invocação assíncrona da função do Lambda do backend

Na integração não proxy do Lambda (personalizada), a função do Lambda do backend é invocada de forma síncrona por padrão. Este é o comportamento desejado para a maioria das operações da API REST. No entanto, alguns aplicativos exigem que o trabalho seja executado de forma assíncrona (como uma operação em lote ou uma operação de longa latência), normalmente por um componente de backend separado. Nesse caso, a função de backend do Lambda é invocada de forma assíncrona, o método da API REST do front-end não retorna o resultado.

É possível configurar a função do Lambda para que uma integração não proxy do Lambda seja invocada de forma assíncrona especificando 'Event' como o [tipo de invocação do](#). Isso é feito da seguinte forma:

Configurar a invocação assíncrona do Lambda no console do API Gateway

Para que todas as invocações sejam assíncronas:

- Em Solicitação de integração, adicione um cabeçalho X-Amz-Invocation-Type com um valor estático de 'Event'.

Para que os clientes decidam se as invocações são assíncronas ou síncronas:

1. Em Solicitação de método, adicione um cabeçalho InvocationType.
2. Em Solicitação de integração, adicione um cabeçalho X-Amz-Invocation-Type com uma expressão de mapeamento de `method.request.header.InvocationType`.
3. Os clientes podem incluir o cabeçalho `InvocationType: Event` em solicitações de API para chamadas assíncronas ou `InvocationType: RequestResponse` para chamadas síncronas.

Configurar a invocação assíncrona do Lambda usando o OpenAPI

Para que todas as invocações sejam assíncronas:

- Adicione o cabeçalho X-Amz-Invocation-Type à seção 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 os clientes decidam se as invocações são assíncronas ou síncronas:

1. Adicione o cabeçalho a seguir em qualquer [objeto de item de caminho do OpenAPI](#).

```
"parameters" : [ {
  "name" : "InvocationType",
  "in" : "header",
  "schema" : {
    "type" : "string"
  }
} ]
```

2. Adicione o cabeçalho X-Amz-Invocation-Type à seção 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. Os clientes podem incluir o cabeçalho `InvocationType: Event` em solicitações de API para chamadas assíncronas ou `InvocationType: RequestResponse` para chamadas síncronas.

Configurar a invocação assíncrona do Lambda usando o CloudFormation

Os modelos CloudFormation a seguir mostram como configurar o `AWS::ApiGateway::Method` para invocações assíncronas.

Para que todas as invocações sejam assíncronas:

```

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 os clientes decidam se as invocações são assíncronas ou síncronas:


```

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'

```

Os clientes podem incluir o cabeçalho `InvocationType: Event` em solicitações de API para chamadas assíncronas ou `InvocationType: RequestResponse` para chamadas síncronas.

Manipular erros do Lambda no API Gateway

Para integrações personalizadas do Lambda, você deve mapear erros retornados pelo Lambda na resposta de integração para respostas de erro HTTP padrão para os seus clientes. Caso contrário, os erros do Lambda serão retornados como respostas 200 OK por padrão, e o resultado não será intuitivo para os usuários da sua API.

Existem dois tipos de erros que o Lambda pode retornar: erros padrão e erros personalizados. Na sua API, você deve lidar com eles de maneira diferente.

Com a integração de proxy do Lambda, o Lambda precisa retornar uma saída com o seguinte formato:

```
{
```

```
"isBase64Encoded" : "boolean",
"statusCode": "number",
"headers": { ... },
"body": "JSON string"
}
```

Nessa saída, `statusCode` é normalmente 4XX para um erro de cliente e 5XX para um erro de servidor. O API Gateway lida com esses erros, mapeando o erro do Lambda para uma resposta de erro HTTP, de acordo com o especificad `statusCode`. Para que o API Gateway transmita o tipo de erro (por exemplo, `InvalidParameterException`) como parte da resposta ao cliente, a função do Lambda deve incluir um cabeçalho (por exemplo, `"X-Amzn-ErrorType": "InvalidParameterException"`) na propriedade `headers`.

Tópicos

- [Manipule erros padrão do Lambda no API Gateway](#)
- [Manipule erros personalizados do Lambda no API Gateway](#)

Manipule erros padrão do Lambda no API Gateway

Um erro do AWS Lambda padrão tem o seguinte formato:

```
{
  "errorMessage": "<replaceable>string</replaceable>",
  "errorType": "<replaceable>string</replaceable>",
  "stackTrace": [
    "<replaceable>string</replaceable>",
    ...
  ]
}
```

Aqui, `errorMessage` é uma expressão de string do erro. O `errorType` é um erro ou tipo de exceção que depende da linguagem. O `stackTrace` é uma lista de expressões de string que mostra o rastreamento de pilha que leva à ocorrência do erro.

Por exemplo, considere a seguinte função do Lambda do JavaScript (Node.js).

```
export const handler = function(event, context, callback) {
  callback(new Error("Malformed input ..."));
};
```

Essa função retorna o seguinte erro padrão do Lambda, contendo `Malformed input ...` como a mensagem de erro:

```
{
  "errorMessage": "Malformed input ...",
  "errorType": "Error",
  "stackTrace": [
    "export const handler (/var/task/index.js:3:14)"
  ]
}
```

Da mesma forma, considere a seguinte função do Lambda Python, que gera um `Exception` com a mesma mensagem de erro `Malformed input ....`

```
def lambda_handler(event, context):
    raise Exception('Malformed input ...')
```

Essa função retorna o seguinte erro padrão do Lambda:

```
{
  "stackTrace": [
    [
      "/var/task/lambda_function.py",
      3,
      "lambda_handler",
      "raise Exception('Malformed input ...')"
    ]
  ],
  "errorType": "Exception",
  "errorMessage": "Malformed input ..."
}
```

Observe que os valores de propriedade `errorType` e `stackTrace` são dependentes da linguagem. O erro-padrão também se aplica a qualquer objeto de erro que seja uma extensão do objeto `Error` ou uma subclasse da classe `Exception`.

Para mapear o erro padrão do Lambda para uma resposta de método, você deve primeiro decidir sobre um código de status HTTP para um determinado erro do Lambda. Depois, defina um padrão de expressão regular na propriedade [selectionPattern](#) do recurso [IntegrationResponse](#) associado ao código de status HTTP especificado. No console do API Gateway,

este `selectionPattern` é indicado como regex de erro do Lambda na seção Resposta de integração, abaixo de cada resposta de integração.

 Note

O API Gateway usa regexes de estilo padrão de Java para o mapeamento de resposta. Para obter mais informações, consulte [Padrão](#) na documentação do Oracle.

Por exemplo, use o comando [put-integration-response](#) indicado abaixo para definir uma nova expressão `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
```

Certifique-se de configurar também o código de erro correspondente (400) na [resposta do método](#). Caso contrário, o API Gateway lançará uma resposta de erro de configuração inválida em tempo de execução.

 Note

No tempo de execução, o API Gateway corresponde a `errorMessage` do erro do Lambda em relação ao padrão da expressão regular na propriedade `selectionPattern`. Se houver uma correspondência, o API Gateway retornará o erro do Lambda como uma resposta HTTP do código de status HTTP correspondente. Se não houver correspondência, o API Gateway retornará o erro como uma resposta padrão ou lançará uma exceção de configuração inválida se não houver uma resposta padrão configurada.

Definir o valor `selectionPattern` como `. *` para uma determinada resposta redefine essa resposta como a resposta padrão. Isso ocorre porque esse padrão de seleção corresponderá a todas as mensagens de erro, incluindo nulo, ou seja, qualquer mensagem de erro não especificado. O mapeamento resultante substitui o mapeamento padrão. Se você usar `.+` como o padrão de seleção para filtrar respostas, saiba que ele pode não corresponder a uma resposta que contém um caractere de nova linha (`\n`).

Para atualizar um valor `selectionPattern` existente usando a AWS CLI, chame a operação [update-integration-response](#) para substituir o valor do caminho `/selectionPattern` pela expressão regex especificada do padrão `Malformed*`.

Para definir a expressão `selectionPattern` usando o console do API Gateway, digite a expressão na caixa de texto Regex de erro do Lambda ao configurar ou atualizar uma resposta de integração de um código de status HTTP especificado.

Manipule erros personalizados do Lambda no API Gateway

Em vez do erro padrão descrito na seção anterior, o AWS Lambda permite retornar um objeto de erro personalizado como uma string JSON. O erro pode ser qualquer objeto JSON válido. Por exemplo, a seguinte função do Lambda do JavaScript (Node.js) retorna um erro 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));  
};
```

Você deve transformar o objeto `myErrorObj` em uma string JSON antes de chamar `callback` para sair da função. Caso contrário, `myErrorObj` será retornado como uma string de `"[object Object]"`. Quando um método da sua API é integrado com a função do Lambda anterior, o API Gateway recebe uma resposta de integração com a seguinte carga:

```
{  
  "errorMessage": "{\"errorType\":\"InternalServerError\",\"httpStatus\":500,  
  \"requestId\":\"e5849002-39a0-11e7-a419-5bb5807c9fb2\",\"trace\":{\"function\":  
  \"abc()\",\"line\":123,\"file\":\"abc.js\"}}"  
}
```

Como ocorre com qualquer resposta de integração, é possível transmitir essa resposta de erro em seu estado inalterado como a resposta do método. Outra opção é fazer com que um modelo de mapeamento transforme a carga útil em um formato diferente. Por exemplo, considere o seguinte modelo de mapeamento de corpo para uma resposta de método do código de status 500:

```
{
  errorMessage: $input.path('$.errorMessage');
}
```

Esse modelo converte o corpo da resposta de integração que contém a string JSON no seguinte corpo de resposta de método. Esse corpo de resposta de método contém o objeto JSON de erro personalizado:

```
{
  "errorMessage" : {
    "errorType" : "InternalServerError",
    "httpStatus" : 500,
    "requestId" : context.awsRequestId,
    "trace" : {
      "function": "abc()",
      "line": 123,
      "file": "abc.js"
    }
  }
};
```

Dependendo dos requisitos da sua API, talvez você precise transmitir algumas das propriedades do erro personalizado, ou todas elas, como parâmetros de cabeçalho da resposta de método. Isso pode ser feito aplicando os mapeamentos de erros personalizados do corpo da resposta de integração aos cabeçalhos da resposta de método.

Por exemplo, a seguinte extensão do OpenAPI define um mapeamento das propriedades `errorMessage.errorType`, `errorMessage.httpStatus`, `errorMessage.trace.function` e `errorMessage.trace` para os cabeçalhos `error_type`, `error_status`, `error_trace_function` e `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"
    },
    ...
  }
}

```

Em tempo de execução, o API Gateway desserializa o parâmetro `integration.response.body` ao executar mapeamentos de cabeçalho. No entanto, essa desserialização aplica-se somente aos mapeamentos de corpo para cabeçalho de respostas de erro personalizado do Lambda, e não a mapeamentos de corpo para corpo usando `$input.body`. Com esses mapeamentos de corpo para cabeçalho de erro personalizado, o cliente recebe os seguintes cabeçalhos como parte da resposta de método, desde que os cabeçalhos `error_status`, `error_trace`, `error_trace_function` e `error_type` estejam declarados na solicitação do método.

```

"error_status": "500",
"error_trace": "{\"function\": \"abc()\", \"line\": 123, \"file\": \"abc.js\"}",
"error_trace_function": "abc()",
"error_type": "InternalServerError"

```

A propriedade `errorMessage.trace` do corpo da resposta de integração é uma propriedade complexa. Ela é mapeada para o cabeçalho `error_trace` como uma string JSON.

Integrações de HTTP para APIs REST no API Gateway

Você pode integrar um método de API com um endpoint HTTP usando a integração de proxy ou a integração personalizada HTTP.

O API Gateway oferece suporte às seguintes portas endpoint: 80, 443 e 1024-65535.

Com a integração de proxy, a configuração é simples. Você só precisa definir o método HTTP e o URI de endpoint HTTP de acordo com os requisitos de backend, se você não estiver preocupado com a codificação de conteúdo nem o armazenamento em cache.

Com a integração personalizada, a configuração é mais complexa. Além das etapas de configuração da integração de proxy, você precisa especificar como os dados da solicitação de entrada serão mapeados para a solicitação de integração e como os dados da resposta de integração resultante serão mapeados para a resposta do método.

Tópicos

- [Configurar integrações de proxy HTTP no API Gateway](#)
- [Configurar integrações personalizadas HTTP no API Gateway](#)

Configurar integrações de proxy HTTP no API Gateway

Para configurar um recurso de proxy com o tipo de integração de proxy HTTP, crie um recurso de API com um parâmetro de caminho voraz (por exemplo, `/parent/{proxy+}`) e integre esse recurso a um endpoint de backend HTTP (por exemplo, `https://petstore-demo-endpoint.execute-api.com/petstore/{proxy}`) no método ANY. O parâmetro de caminho voraz deve estar no final do caminho do recurso.

Como no caso de um recurso não proxy, é possível configurar um recurso de proxy com a integração de proxy HTTP usando o console do API Gateway, importando um arquivo de definição do OpenAPI ou chamando diretamente a API REST do API Gateway. Para obter instruções detalhadas sobre como usar o console do API Gateway para configurar um recurso de proxy com a integração HTTP, consulte [Tutorial: Crie uma API REST com uma integração de proxy de HTTP](#).

O seguinte arquivo de definição do OpenAPI mostra um exemplo de uma API com um recurso de proxy que está integrado ao site [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"  
      }  
    }  
  }  
}
```

Neste exemplo, uma chave de cache é declarada no parâmetro de caminho `method.request.path.proxy` do recurso de proxy. Esta é a configuração padrão quando você cria a API usando o console do API Gateway. O caminho base da API (`/test`, correspondente a um estágio) é mapeado para a página PetStore do site (`/petstore`). A solicitação de integração única espelha o site PetStore inteiro usando a variável de caminho voraz da API e o método genérico ANY. Os exemplos a seguir ilustram esse espelhamento.

- Defina **ANY** como **GET** e **{proxy+}** como **pets**

Solicitação de método iniciada no front-end:

```
GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets HTTP/1.1
```

Solicitação de integração enviada ao backend:

```
GET http://petstore-demo-endpoint.execute-api.com/petstore/pets HTTP/1.1
```

As instâncias de tempo de execução do método ANY e o recurso de proxy são ambos válidos. A chamada retorna uma resposta 200 OK com a carga que contém o primeiro lote de animais de estimação, conforme retornado do backend.

- Defina **ANY** como **GET** e **{proxy+}** como **pets?type=dog**

```
GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets?type=dog  
HTTP/1.1
```

Solicitação de integração enviada ao backend:

```
GET http://petstore-demo-endpoint.execute-api.com/petstore/pets?type=dog HTTP/1.1
```

As instâncias de tempo de execução do método ANY e o recurso de proxy são ambos válidos. A chamada retorna uma resposta 200 OK com a carga que contém o primeiro lote de cães especificados, conforme retornado do backend.

- Defina **ANY** como **GET** e **{proxy+}** como **pets/{petId}**

Solicitação de método iniciada no front-end:

```
GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets/1 HTTP/1.1
```

Solicitação de integração enviada ao backend:

```
GET http://petstore-demo-endpoint.execute-api.com/petstore/pets/1 HTTP/1.1
```

As instâncias de tempo de execução do método ANY e o recurso de proxy são ambos válidos. A chamada retorna uma resposta 200 OK com a carga que contém o animal de estimação especificado, conforme retornado do backend.

- Defina **ANY** como **POST** e **{proxy+}** como **pets**

Solicitação de método iniciada no front-end:

```
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
}
```

Solicitação de integração enviada ao 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
}
```

```
}
```

As instâncias de tempo de execução do método ANY e o recurso de proxy são ambos válidos. A chamada retorna uma resposta 200 OK com a carga que contém o animal de estimação recém-criado, conforme retornado do backend.

- Defina **ANY** como **GET** e **{proxy+}** como **pets/cat**

Solicitação de método iniciada no front-end:

```
GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/pets/cat
```

Solicitação de integração enviada ao backend:

```
GET http://petstore-demo-endpoint.execute-api.com/petstore/pets/cat
```

A instância de runtime do caminho do recurso de proxy não corresponde a um endpoint de backend, e a solicitação resultante é inválida. Como resultado, uma resposta 400 Bad Request é retornada com a seguinte mensagem de erro.

```
{
  "errors": [
    {
      "key": "Pet2.type",
      "message": "Missing required field"
    },
    {
      "key": "Pet2.price",
      "message": "Missing required field"
    }
  ]
}
```

- Defina **ANY** como **GET** e **{proxy+}** como **null**

Solicitação de método iniciada no front-end:

```
GET https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test
```

Solicitação de integração enviada ao backend:

```
GET http://petstore-demo-endpoint.execute-api.com/petstore/pets
```

O recurso direcionado é o pai do recurso de proxy, mas a instância de tempo de execução do método ANY não está definida na API nesse recurso. Como resultado, essa solicitação GET retorna uma resposta 403 Forbidden com a mensagem de erro Missing Authentication Token retornada pelo API Gateway. Se a API expõe o método ANY ou GET no recurso pai, (/), a chamada retorna uma resposta 404 Not Found com a mensagem Cannot GET /petstore conforme retornada do backend.

Para qualquer solicitação de cliente, se a URL do endpoint de destino for inválida ou se o verbo HTTP for válido, mas não tiver suporte, o backend retornará uma resposta 404 Not Found. Para um método HTTP sem suporte, uma resposta 403 Forbidden é retornada.

Configurar integrações personalizadas HTTP no API Gateway

Com a integração HTTP personalizada, também conhecida como integração sem proxy, você tem mais controle de quais dados passar entre um método de API e uma integração de API e como passar os dados. Isso é feito pelo mapeamento de dados.

Como parte da configuração da solicitação do método, defina a propriedade [requestParameters](#) em um recurso [Method](#). Isso declara quais parâmetros da solicitação do método, que são provisionados pelo cliente, devem ser mapeados para os parâmetros da solicitação de integração ou para as propriedades do corpo aplicáveis antes de serem enviados para o backend. Depois, como parte da configuração da solicitação de integração, defina a propriedade [requestParameters](#) no recurso [Integration](#) correspondente para especificar os mapeamentos entre parâmetros. Você também define a propriedade [requestTemplates](#) para especificar modelos de mapeamento, um para cada tipo de conteúdo com suporte. Os modelos de mapeamento mapeiam o corpo ou os parâmetros da solicitação do método ao corpo da solicitação da integração.

De forma similar, como parte da configuração da resposta do método, você define a propriedade [responseParameters](#) no recurso [MethodResponse](#). Isso declara quais parâmetros de resposta do método, a serem enviados ao cliente, devem ser mapeados dos parâmetros da resposta de integração ou de determinadas propriedades do corpo aplicáveis que foram retornadas do backend. Depois, você configura o [selectionPattern](#) para escolher uma resposta de integração com base na resposta do backend. Para uma integração HTTP sem proxy, isso é uma expressão regular. Por exemplo, para mapear todos os códigos de status de resposta HTTP 2xx de um endpoint HTTP para esse mapeamento de saída, use `2\d{2}`.

 Note

O API Gateway usa regexes de estilo padrão de Java para o mapeamento de resposta. Para obter mais informações, consulte [Padrão](#) na documentação do Oracle.

Depois, como parte da configuração da resposta de integração, defina a propriedade [responseParameters](#) no recurso [IntegrationResponse](#) correspondente para especificar os mapeamentos entre parâmetros. Você também define o mapa [responseTemplates](#) para especificar modelos de mapeamento, um para cada tipo de conteúdo com suporte. Os modelos de mapeamento mapeiam os parâmetros da resposta de integração ou as propriedades do corpo da resposta de integração ao corpo da resposta do método.

Para obter mais informações sobre como configurar modelos de mapeamento, consulte [Transformações de dados para APIs REST no API Gateway](#).

Transmitir a resposta para suas integrações de proxy no API Gateway

É possível configurar sua integração de proxy para controlar como o API Gateway exibe sua resposta de integração. Por padrão, o API Gateway espera receber a resposta completa antes de iniciar a transmissão. No entanto, se você definir o modo de transferência de resposta da sua integração como `STREAM`, o API Gateway vai esperar uma resposta ser completamente calculada antes de enviá-la ao cliente. O streaming de resposta funciona para todos os tipos de endpoints da API REST.

Use o streaming de resposta para os seguintes casos de uso:

- Reduzir o tempo até o primeiro byte (TTFB) para aplicações de IA generativa, como chatbots.
- Transmitir arquivos grandes de imagem, vídeo ou música sem utilizar um URL pré-assinado do S3.
- Realizar operações de longa duração enquanto relata o progresso incremental, como eventos enviados pelo servidor (SSE).
- Exceder o limite de carga útil de resposta de 10 MB do API Gateway.
- Exceder o limite de tempo limite de 29 segundos do API Gateway sem solicitar aumento do tempo limite de integração.
- Receber uma carga útil binária sem configurar os tipos de mídia binária.

Considerações sobre streaming de carga útil de resposta

As seguintes considerações podem afetar o uso de streaming de carga útil de resposta:

- Você só pode usar streaming de carga útil de resposta para os tipos de integração `HTTP_PROXY` ou `AWS_PROXY`. Isso inclui integrações de proxy do Lambda e integrações privadas que utilizem integrações `HTTP_PROXY`.
- A configuração padrão do modo de transferência é `BUFFERED`. Para usar o streaming de resposta, você deve alterar o modo de transferência de resposta para `STREAM`.
- O streaming de resposta só aceita APIs REST.
- A solicitação de streaming não é aceita.
- Você pode transmitir sua resposta por até 15 minutos.
- Seus fluxos estão sujeitos a tempos limite de inatividade. No caso de endpoints regionais ou privados, o tempo limite é de 5 minutos. Em relação a endpoints otimizados para borda, o tempo limite é de 30 segundos.

- Se você usar streaming de resposta para uma API REST regional com sua própria distribuição do CloudFront, poderá atingir um tempo limite de inatividade superior a 30 segundos aumentando o tempo limite de resposta da sua distribuição do CloudFront. Para acessar mais informações, consulte [Tempo limite de resposta](#).
- Quando o modo de transferência de resposta está definido como `STREAM`, o API Gateway não comporta recursos que exijam o armazenamento em buffer de toda a resposta da integração. Por esse motivo, os seguintes recursos não comportam o streaming de respostas:
 - Cache de endpoints
 - Codificação de conteúdo. Se você quiser compactar sua resposta de integração, faça isso em sua integração.
 - Transformação de resposta com VTL
- Em cada resposta de streaming, os primeiros 10 MB de carga útil de resposta não estão sujeitos a nenhuma restrição de largura de banda. Os dados de carga útil de resposta que excedem 10 MB ficam restritos a 2 MB/s.
- Quando a conexão entre o cliente e o API Gateway ou entre o API Gateway e o Lambda é fechada devido ao tempo limite, a função do Lambda pode continuar sendo executada. Para acessar mais informações, consulte [Configurar tempo limite das funções do Lambda](#).
- O streaming de resposta gera um custo. Para obter mais informações, consulte [Preços do API Gateway](#).

Configurar uma integração de proxy HTTP com streaming de resposta de carga útil no API Gateway

Ao configurar o streaming de carga útil de resposta, você especifica o modo de transferência de resposta na solicitação de integração do seu método. Você precisa definir essas configurações na solicitação de integração a fim de controlar como o API Gateway se comporta antes e durante a resposta de integração. Quando você utiliza o streaming de resposta, pode configurar o tempo limite de integração como até 15 minutos.

Quando você usa o streaming de resposta de carga útil com uma integração `HTTP_PROXY`, o API Gateway não envia o código de status da resposta HTTP nem o cabeçalho de resposta HTTP até receber integralmente todos os cabeçalhos.

Configurar uma integração de proxy HTTP com streaming de resposta de carga útil

O procedimento a seguir mostra como importar uma nova API com o `responseTransferMode` definido como `STREAM`. Se você tiver uma API de integração existente e quiser modificar o

`responseTransferMode`, consulte [the section called “Atualizar o modo de transferência de resposta para uma integração de proxy HTTP”](#).

Console de gerenciamento da AWS

Como configurar uma integração de proxy HTTP com streaming de resposta de carga útil

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST.
3. Selecione Criar recurso.
4. Em Resource Name (Nome do recurso), insira **streaming**.
5. Selecione Criar recurso.
6. Com o recurso /streaming selecionado, escolha Criar método.
7. Em Tipo de método, selecione ANY.
8. Em Tipo de integração, escolha HTTP.
9. Selecione Integração de proxy HTTP.
10. Em Modo de transferência de resposta, selecione Fluxo.
11. Em Método HTTP, escolha um método.
12. Em URL do endpoint, insira um endpoint de integração. Escolha um endpoint que produza uma grande carga útil para ser transmitida de volta para você.
13. Escolha Criar método.

Depois de criar o método, implante a API.

Para implantar sua API

1. Escolha Implantar API.
2. Em Estágio, selecione Novo estágio.
3. Em Stage name (Nome do estágio), insira **prod**.
4. (Opcional) Em Description (Descrição), insira uma descrição.
5. Escolha Implantar.

AWS CLI

Como criar uma API com streaming de resposta de carga útil

1. Copie o arquivo Open API a seguir e, depois, salve-o como `ResponseStreamDemoSwagger.yaml`. Nesse arquivo, `responseTransferMode` está definido como `STREAM`. O endpoint de integração está definido como `https://example.com`, mas recomendamos que você o modifique para um endpoint que produza uma grande carga útil a ser transmitida de volta para você.

```
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. Utilize o seguinte comando `import-rest-api` para importar sua definição de OpenAPI:

```
aws apigateway import-rest-api \
  --body 'fileb://~/ResponseStreamDemoSwagger.yaml' \
  --parameters endpointConfigurationTypes=REGIONAL \
  --region us-west-1
```

3. Utilize o seguinte comando `create-deployment` para implantar a nova API em um estágio:

```
aws apigateway create-deployment \
  --rest-api-id a1b2c3 \
  --stage-name prod \
  --region us-west-1
```

Atualizar o modo de transferência de resposta para uma integração de proxy HTTP

O procedimento a seguir mostra como atualizar o modo de transferência de resposta para uma integração de proxy HTTP.

Console de gerenciamento da AWS

Como atualizar o modo de transferência de resposta para uma integração de proxy HTTP

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST.
3. Escolha um método.
4. Na guia Solicitação de integração, em Configurações de solicitação de integração, selecione Editar.
5. Em Modo de transferência de resposta, selecione Fluxo.
6. Escolha Salvar.

Depois de atualizar o método, implante a API.

Para implantar sua API

1. Escolha Implantar API.
2. Em Estágio, selecione Novo estágio.
3. Em Stage name (Nome do estágio), insira **prod**.
4. (Opcional) Em Description (Descrição), insira uma descrição.
5. Escolha Implantar.

AWS CLI

O comando `update-integration` a seguir atualiza o modo de transferência de uma integração de BUFFERED para STREAM. Para todas as APIs existentes, o modo de transferência de resposta para todas as integrações está definido como BUFFERED.

```
aws apigateway update-integration \  
  --rest-api-id a1b2c3 \  
  --resource-id aaa111 \  
  --method-name GET
```

```
--http-method GET \  
--patch-operations "op='replace',path='/responseTransferMode',value=STREAM" \  
--region us-west-1
```

Será necessário implantar a API novamente para que as alterações sejam implementadas. Se você personalizou o tempo limite de integração, esse valor de tempo limite será removido, pois o API Gateway transmite sua resposta por até 5 minutos.

O seguinte comando `update-integration` atualiza o modo de transferência de uma integração de STREAM para 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
```

Será necessário implantar a API novamente para que as alterações sejam implementadas.

Configurar uma integração de proxy do Lambda com streaming de resposta da carga útil no API Gateway

É possível realizar o streaming da resposta de uma função do Lambda para melhorar a performance do tempo até o primeiro byte (TTFB) e enviar respostas parciais de volta ao cliente assim que elas se tornam disponíveis. O API Gateway exige que você use a API [InvokeWithResponseStream](#) do Lambda para invocar sua função do Lambda. O API Gateway transmite um objeto de evento para a função do Lambda. A função do Lambda do backend analisa os dados da solicitação recebida para determinar a resposta que ela retorna. Para o API Gateway transmitir a saída do Lambda, a função do Lambda deve gerar o [formato](#) exigido pelo API Gateway.

Diferenças nas integrações de proxy do Lambda entre o modo de transferência de fluxo e resposta em buffer

A lista a seguir descreve as diferenças entre uma integração do proxy do Lambda e uma integração do proxy do Lambda para streaming de resposta:

- O API Gateway utiliza a API [InvokeWithResponseStream](#) para invocar a integração do proxy do Lambda para streaming de respostas. Isso gera um URI diferente, que é o seguinte:

```
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
```

Esse ARN utiliza uma data diferente para a versão da API e uma ação de serviço diferente em comparação à integração do proxy do Lambda.

Se você utiliza o console do API Gateway para streaming de respostas, o console usa o URI correto para você.

- Em uma integração do proxy do Lambda, o API Gateway envia a resposta ao cliente somente depois de receber a resposta completa do Lambda. Em uma integração do proxy do Lambda para streaming de resposta, o API Gateway inicia o fluxo de carga útil depois de receber os metadados e o delimitador válidos do Lambda.
- A integração do proxy do Lambda para streaming de respostas utiliza o mesmo formato de entrada que o da integração de proxy, mas exige um formato de saída diferente.

Formato de integração do proxy do Lambda para streaming de resposta

Quando o API Gateway invoca uma função do Lambda com streaming de resposta, o formato de entrada é igual ao formato de entrada de uma função do Lambda para integração de proxy. Para obter mais informações, consulte [the section called “Formato de entrada de uma função do Lambda para integração de proxy”](#).

Quando o Lambda transmite uma resposta para o API Gateway, ela deve seguir o formato a seguir. Esse formato utiliza um delimitador para separar o JSON dos metadados e a carga útil bruta. Nesse caso, os dados da carga útil são transmitidos no decorrer do streaming realizado pela função de streaming do Lambda:

```
{
  "headers": {"headerName": "headerValue", ...},
  "multiValueHeaders": { "headerName": ["headerValue", "headerValue2", ...], ... },
  "cookies" : ["cookie1", "cookie2"],
  "statusCode": httpStatusCode
}<DELIMITER>PAYLOAD1 | PAYLOAD2 | PAYLOAD3
```

Na saída:

- As chaves `headers`, `multiValueHeaders`, `cookies` e `statusCode` poderão ficar sem especificação se nenhum cabeçalho de resposta extra precisar ser exibido.

- A chave `headers` só pode conter cabeçalhos de valor único.
- A saída espera que os cabeçalhos contêm `Transfer-Encoding: chunked` ou `Content-length: number`. Se sua função não exibir nenhum desses cabeçalhos, o API Gateway anexará `Transfer-Encoding: chunked` ao cabeçalho de resposta.
- A chave `multiValueHeaders` pode conter cabeçalhos de vários valores e cabeçalhos de valor único. Você pode usar a chave `multiValueHeaders` para especificar todos os seus cabeçalhos extras, incluindo os de valor único.
- Se você especificar valores para `headers` e `multiValueHeaders`, o API Gateway os mesclará em uma única lista. Se o mesmo de chave/valor for especificado em ambos, somente os valores de `multiValueHeaders` aparecerão na lista mesclada.
- Os metadados devem ser JSON válido. Somente as chaves `headers`, `multiValueHeaders`, `cookies` e `statusCode` são aceitas.
- Você deve fornecer um delimitador após o JSON dos metadados. O delimitador deve ter 8 bytes nulos e deve aparecer nos primeiros 16 KB de dados do fluxo.
- O API Gateway não exige um formato específico para a carga útil de resposta do método.

Se você estiver usando um URL da função para transmitir sua função do Lambda, deverá modificar a entrada e a saída da sua função do Lambda para atender a esses requisitos.

Se a saída da função do Lambda não atender aos requisitos desse formato, o API Gateway ainda poderá invocar sua função do Lambda. A tabela a seguir mostra as combinações das configurações de solicitação de integração da API e do código da função do Lambda que comportam o API Gateway. Isso inclui combinações aceitas pelo modo de transferência de resposta em buffer.

Modo de transferência de resposta	O código da função segue o formato exigido	API de invocação do Lambda	Aceito pelo API Gateway.
Fluxo	Sim	InvokeWithResponseStream	Sim. O API Gateway transmite sua resposta.
Fluxo	Não	InvokeWithResponseStream	Não. O API Gateway invocará a função do Lambda e exibirá uma

Modo de transferência de resposta	O código da função segue o formato exigido	API de invocação do Lambda	Aceito pelo API Gateway.
			resposta de erro de 500.
Fluxo	Sim	Invocar	Não. O API Gateway não aceita essa configuração de integração.
Fluxo	Não	Invocar	Não. O API Gateway não aceita essa configuração de integração.
Armazenado em buffer	Sim	InvokeWithResponseStream	Não. O API Gateway não aceita essa configuração de integração.
Armazenado em buffer	Não	InvokeWithResponseStream	Não. O API Gateway não aceita essa configuração de integração.
Armazenado em buffer	Sim	Invocar	O API Gateway exibe os cabeçalhos HTTP e o código de status, mas não o corpo da resposta.

Modo de transferência de resposta	O código da função segue o formato exigido	API de invocação do Lambda	Aceito pelo API Gateway.
Armazenado em buffer	Não	Invocar	Sim. Esta é uma integração do proxy do Lambda. Para acessar mais informações, consulte Integração do proxy do Lambda .

Configurar uma integração de proxy do Lambda com streaming de resposta da carga útil no API Gateway

Quando você configura o streaming de carga útil de resposta, especifica o modo de transferência na resposta de integração do seu recurso. Você precisa definir essas configurações na solicitação de integração a fim de controlar como o API Gateway se comporta antes e durante a resposta de integração.

Exemplos de função do Lambda para streaming de resposta

A sua função do Lambda deve seguir o [the section called “Formato de integração do proxy do Lambda para streaming de resposta”](#). Recomendamos que você use um dos três exemplos de funções do Lambda para testar o streaming de respostas. Ao criar a função do Lambda, faça o seguinte:

- Forneça um tempo limite adequado para sua função. Recomendamos que você configure um tempo limite de pelo menos 1 minuto para saber mais sobre o streaming de respostas. Ao criar seus recursos de produção, garanta que o tempo limite da função do Lambda abranja todo o ciclo de solicitações. Para acessar mais informações, consulte [Configurar tempo limite das funções do Lambda](#).
- Use o runtime mais recente do Node.js.
- Use uma região onde o streaming de respostas do Lambda esteja disponível.

Using `HttpResponseStream.from`

O exemplo de código a seguir transmite os objetos e cargas úteis de metadados JSON de volta para o cliente utilizando o método `awslambda.HttpResponseStream()` sem usar o método de pipeline. Não é necessário criar o delimitador. Para acessar mais informações, consulte [Escrever funções do Lambda habilitadas para streaming de resposta](#).

```
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

O Lambda recomenda que, ao escrever funções habilitadas para streaming de resposta, você use o decorador `awslambda.streamifyResponse()` fornecido pelos runtimes nativos do Node.js e o método `pipeline()`. Quando você utiliza o método de pipeline, não precisa criar o delimitador. O Lambda faz isso para você. Para acessar mais informações, consulte [Escrever funções do Lambda habilitadas para streaming de resposta](#).

O exemplo de código a seguir transmite os objetos e cargas úteis de metadados JSON de volta para o 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

O exemplo de código a seguir transmite os objetos e cargas úteis de metadados JSON de volta para o cliente sem utilizar o método `awslambda.HttpResponseStream()`. Sem o método `awslambda.HttpResponseStream()`, você deve incluir um delimitador de 8 bytes nulos entre os metadados e a 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();
});
```

Criar uma integração de proxy do Lambda com streaming de resposta da carga útil

O procedimento a seguir mostra como criar uma integração do proxy do Lambda com streaming da resposta da carga útil. Use o exemplo da função do Lambda ou crie a sua própria.

Console de gerenciamento da AWS

Como criar uma integração do proxy do Lambda com streaming de resposta da carga útil

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST.
3. Selecione Criar recurso.
4. Em Resource Name (Nome do recurso), insira **streaming**.
5. Selecione Criar recurso.
6. Com o recurso /streaming selecionado, escolha Criar método.
7. Em Tipo de método, selecione ANY.
8. Em Tipo de integração, escolha Lambda.
9. Selecione Integração de proxy do Lambda.
10. Em Modo de transferência de resposta, selecione Fluxo.
11. Em Função do Lambda, escolha o nome da sua função do Lambda.

O console do API Gateway utiliza automaticamente a API [InvokeWithResponseStream](#) para invocar a função do Lambda. Você é responsável por escrever uma função do Lambda habilitada para streaming de resposta. Para ver um exemplo, consulte [the section called “Exemplos de função do Lambda para streaming de resposta”](#).

12. Escolha Criar método.

Depois de criar o método, implante a API.

Para implantar sua API

1. Escolha Implantar API.
2. Em Estágio, selecione Novo estágio.
3. Em Stage name (Nome do estágio), insira **prod**.
4. (Opcional) Em Description (Descrição), insira uma descrição.
5. Escolha Implantar.

AWS CLI

O procedimento a seguir mostra como importar uma nova API com o `responseTransferMode` definido como `STREAM`. Se você tiver uma API de integração existente e quiser modificar o `responseTransferMode`, consulte [the section called “Atualizar o modo de transferência de resposta para uma integração de proxy do Lambda”](#).

Como criar uma API com streaming de resposta de carga útil

1. Copie o arquivo Open API a seguir e, depois, salve-o como `ResponseStreamDemoSwagger.yaml`. Nesse arquivo, `responseTransferMode` é definido como `STREAM`, e o URI de integração é definido como `arn:aws:apigateway:us-west-1:lambda:path/2021-11-15/functions/arn:aws:lambda:us-west-1:11112223333:function:my-function-name/response-streaming-invocations`.

Substitua o nome da função de `my-function` por uma função habilitada para streaming e substitua as credenciais por um perfil do IAM que tenha políticas que permitem que o serviço `apigateway` invoque funções do 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"
```

Em vez de fornecer um perfil do IAM para recuperar credenciais, você pode utilizar o comando `add-permission` para o Lambda adicionar permissões baseadas em recurso.

2. Utilize o seguinte comando `import-rest-api` para importar sua definição de OpenAPI:

```
aws apigateway import-rest-api \
  --body 'fileb://~/ResponseStreamDemoSwagger.yaml' \
  --parameters endpointConfigurationTypes=REGIONAL \
  --region us-west-1
```

3. Utilize o seguinte comando `create-deployment` para implantar a nova API em um estágio:

```
aws apigateway create-deployment \
  --rest-api-id a1b2c2 \
  --stage-name prod \
  --region us-west-1
```

Atualizar o modo de transferência de resposta para uma integração de proxy do Lambda

O procedimento a seguir mostra como atualizar o modo de transferência de resposta para uma integração de proxy do Lambda. Ao alterar o modo de transferência de resposta para streaming, atualize sua função do Lambda para que ela atenda aos requisitos de streaming de resposta. Use o exemplo da função do Lambda ou crie a sua própria.

Console de gerenciamento da AWS

Como atualizar o modo de transferência de resposta para uma integração do proxy do Lambda

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST.
3. Escolha um método.
4. Na guia Solicitação de integração, em Configurações de solicitação de integração, selecione Editar.
5. Em Modo de transferência de resposta, selecione Fluxo.
6. Em Função do Lambda, escolha o nome da sua função do Lambda.
7. Escolha Salvar.

Depois de atualizar o método, implante a API.

Para implantar sua API

1. Escolha Implantar API.
2. Em Estágio, selecione Novo estágio.
3. Em Stage name (Nome do estágio), insira **prod**.
4. (Opcional) Em Description (Descrição), insira uma descrição.
5. Escolha Implantar.

AWS CLI

1. Atualize a função do Lambda para que seja habilitada para streaming.
2. Use o seguinte comando da AWS CLI para atualizar o URI de integração e o modo de transferência de resposta da sua integração:

```
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\"}]\" \\\n--region us-west-1
```

3. Implante a API novamente para que as alterações entrem em vigor.

Solucionar problemas com o streaming de respostas no API Gateway

As diretrizes de solução de problemas a seguir podem ajudar a resolver problemas com suas APIs que utilizam streaming de resposta.

Solução de problemas gerais

Você pode usar o [TestInvokeMethod](#) a guia de teste do console para testar a resposta do fluxo. As seguintes considerações podem afetar o uso da invocação de teste para streaming de resposta:

- Quando você testa seu método, o API Gateway armazena em buffer sua carga útil de resposta transmitida. Depois que qualquer uma das condições a seguir for atendida, o API Gateway exibirá uma resposta única contendo a carga útil armazenada em buffer:
 - A solicitação está completa
 - 35 segundos se passaram
 - Mais de 1 MB de carga útil de resposta foi armazenada em buffer
- Se transcorrerem mais de 35 segundos antes que seu método exiba um status de resposta HTTP e todos os cabeçalhos, o status da resposta exibido em TestInvokeMethod será 0.
- O API Gateway não produz nenhum log de execução.

Depois de implantar sua API, você pode testar a resposta do fluxo utilizando um comando curl. É recomendável utilizar a opção `-i` de incluir cabeçalhos da resposta de protocolo na saída. Para ver os dados de resposta à medida que eles chegam, utilize a opção `--no-buffer`.

Solucionar problemas de cURL

Se você estiver testando uma integração e receber o erro curl: (18) transfer closed with outstanding read data remaining, verifique se o tempo limite da integração é longo o suficiente. Se você estiver usando uma função do Lambda, precisará atualizar o tempo limite de resposta dela. Para acessar mais informações, consulte [Configurar tempo limite das funções do Lambda](#).

Solução de problemas no uso do registro em log do acesso

Você pode usar logs de acesso para o estágio da API REST a fim de registrar em log e solucionar problemas em seu fluxo de resposta. Além das variáveis existentes, você pode usar as seguintes variáveis de log de acesso:

`$context.integration.responseTransferMode`

O modo de transferência de resposta da integração. Ele pode ser BUFFERED ou STREAMED.

`$context.integration.timeToAllHeaders`

O tempo entre o momento em que o API Gateway estabelece a conexão de integração e o momento em que recebe todos os cabeçalhos de resposta de integração do cliente.

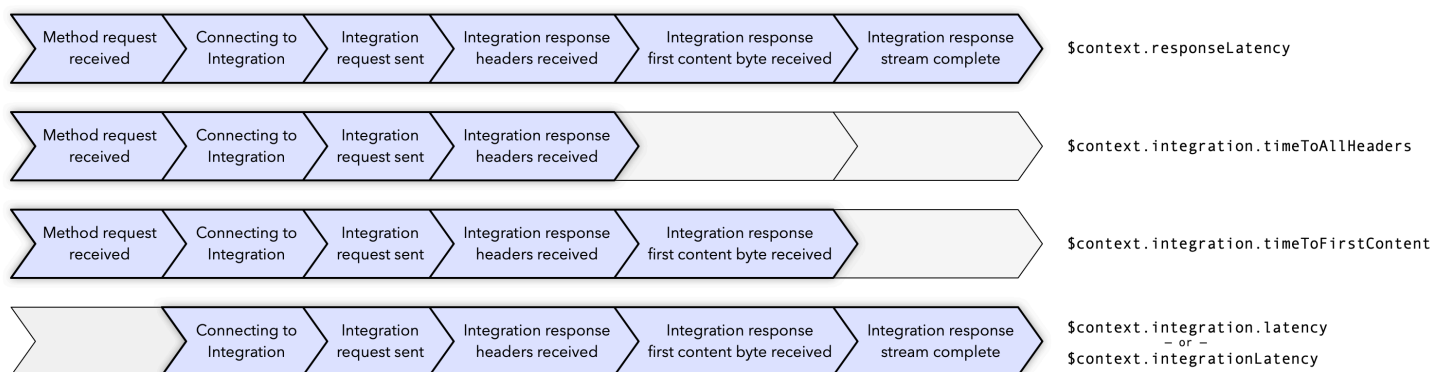
`$context.integration.timeToFirstContent`

O tempo entre o momento em que o API Gateway estabelece a conexão de integração e o momento em que recebe os primeiros bytes de conteúdo.

`$context.integration.latency` ou `$context.integrationLatency`

O momento em que o API Gateway estabelece a conexão de integração e o momento em que o fluxo de resposta de integração é concluído.

A figura a seguir mostra como essas variáveis de log de acesso representam diferentes componentes de um fluxo de resposta.



Para obter mais informações sobre os logs de acesso, consulte [the section called “Logs do CloudWatch”](#). Você também pode usar o X-Ray para monitorar seu fluxo de resposta. Para obter mais informações, consulte [the section called “X-Ray”](#).

Integrações privadas para APIs REST no API Gateway

Utilize uma integração privada para expor seus recursos HTTP/HTTPS em uma Amazon VPC para acesso por clientes fora da VPC. Isso estende o acesso a seus recursos privados da VPC além dos limites da VPC. Você pode controlar o acesso à sua API usando qualquer um dos [métodos de autorização](#) compatíveis com o API Gateway.

Para criar uma integração privada, primeiro crie um link de VPC. O API Gateway aceita links da VPC V2 para APIs REST. Os links da VPC V2 permitem criar integrações privadas que conectam sua API REST aos Application Load Balancers sem utilizar um Network Load Balancer. O uso de um Application Load Balancer permite que você se conecte às aplicações baseadas em contêineres do Amazon ECS e a muitos outros backends. Os links da VPC V1 são considerados um tipo de integração legado. Embora sejam aceitos no API Gateway, recomendamos que você não crie outros links da VPC V1.

Considerações

As considerações a seguir podem afetar o uso de integrações privadas:

- Todos os recursos devem ser de propriedade da mesma Conta da AWS. Isso inclui o balanceador de carga, o link de VPC e a API REST.
- Por padrão, o tráfego de integração privada usa o protocolo HTTP. Para usar HTTPS, especifique um [uri](#) que contenha um nome de servidor seguro, como `https://example.com:443/test`.
- Em uma integração privada, o API Gateway inclui a parte do [estágio](#) do endpoint da API na solicitação para seus recursos de backend. Por exemplo, se você solicitar o estágio `test` de uma API, o API Gateway incluirá `test/path` na solicitação para sua integração privada. Para remover o nome do estágio da solicitação para os seus recursos de backend, use o [mapeamento de parâmetros](#) para criar uma substituição para a variável `$context.requestOverride.path`.
- Integrações privadas com AWS Cloud Map não são aceitas.

Tópicos

- [Configurar links da VPC V2 no API Gateway](#)
- [Configurar uma integração privada](#)
- [Integração privada utilizando links da VPC V1 \(legado\)](#)

Configurar links da VPC V2 no API Gateway

Os links de VPC permitem criar integrações privadas que conectem suas rotas de API a recursos privados em uma VPC, como Application Load Balancers ou aplicações baseadas em contêiner do Amazon ECS. A integração privada usa um link de VPC V2 para encapsular conexões entre o API Gateway e os recursos de VPC desejados. Você pode reutilizar links da VPC em diferentes recursos e APIs.

Quando um link de VPC é criado, o API Gateway cria e gerencia [interfaces de rede elásticas](#) para o link de VPC V2 em sua conta. Esse processo pode levar alguns minutos. Quando um link de VPC V2 está pronto para uso, seu estado faz a transição de PENDING para AVAILABLE.

Note

Se nenhum tráfego for enviado pelo link da VPC por 60 dias, ele se tornará INACTIVE. Quando um link de VPC está em um estado INACTIVE, o API Gateway exclui todas as interfaces de rede do link de VPC. Isso faz com que as solicitações de API que dependem do link da VPC falhem. Se as solicitações de API forem retomadas, o API Gateway provisionará as interfaces de rede novamente. Pode levar alguns minutos para criar as interfaces de rede e reativar o link da VPC. Você pode usar o status do link da VPC para monitorar o estado do link da VPC.

Criar um link da VPC V2 usando a AWS CLI

Para criar um link de VPC V2, todos os recursos envolvidos devem pertencer à mesma conta da AWS. O comando [create-vpc-link](#) indicado abaixo cria um link de VPC:

```
aws apigatewayv2 create-vpc-link --name MyVpcLink \  
  --subnet-ids subnet-aaaa subnet-bbbb \  
  --security-group-ids sg1234 sg5678
```

Note

Os links da VPC V2 são imutáveis. Depois de criar um link de VPC V2, você não poderá alterar suas sub-redes ou grupos de segurança.

Excluir um link da VPC V2 usando a AWS CLI

O comando [delete-vpc-link](#) indicado a seguir exclui um link de VPC.

```
aws apigatewayv2 delete-vpc-link --vpc-link-id abcd123
```

Disponibilidade por região

Os links da VPC V2 são aceitos nas seguintes regiões e zonas de disponibilidade:

Nome da região	Região	Zonas de disponibilidade compatíveis
Leste dos EUA (Ohio)	us-east-2	use2-az1, use2-az2, use2-az3
Leste dos EUA (Norte da Virgínia)	us-east-1	use1-az1, use1-az2, use1-az4, use1-az5, use1-az6
Oeste dos EUA (N. da Califórnia)	us-west-1	usw1-az1, usw1-az3
Oeste dos EUA (Oregon)	us-west-2	usw2-az1, usw2-az2, usw2-az3, usw2-az4
Ásia-Pacífico (Hong Kong)	ap-east-1	ape1-az2, ape1-az3
Ásia-Pacífico (Mumbai)	ap-south-1	aps1-az1, aps1-az2, aps1-az3
Ásia-Pacífico (Seul)	ap-northeast-2	apne2-az1, apne2-az2, apne2-az3
Ásia-Pacífico (Singapura)	ap-southeast-1	apse1-az1, apse1-az2, apse1-az3

Nome da região	Região	Zonas de disponibilidade compatíveis
Ásia-Pacífico (Sydney)	ap-southeast-2	apse2-az1, apse2-az2, apse2-az3
Ásia-Pacífico (Tóquio)	ap-northeast-1	apne1-az1, apne1-az2, apne1-az4
Canadá (Central)	ca-central-1	cac1-az1, cac1-az2
Europa (Frankfurt)	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 (Paris)	eu-west-3	euw3-az1, euw3-az3
Europa (Espanha)	eu-south-2	eus2-az1, eus2-az2, eus2-az3
Europa (Estocolmo)	eu-north-1	eun1-az1, eun1-az2, eun1-az3
Oriente Médio (Barém)	me-south-1	mes1-az1, mes1-az2, mes1-az3
América do Sul (São Paulo)	sa-east-1	sae1-az1, sae1-az2, sae1-az3

Nome da região	Região	Zonas de disponibilidade compatíveis
AWS GovCloud (Oeste dos EUA)	us-gov-west-1	usgw1-az1, usgw1-az2, usgw1-az3

Configurar uma integração privada

Para criar uma integração privada com um Application Load Balancer ou Network Load Balancer, crie uma integração de proxy HTTP, especifique o [link da VPC V2](#) a ser usado e forneça o ARN de um Network Load Balancer ou um Application Load Balancer. Por padrão, o tráfego de integração privada usa o protocolo HTTP. Para usar HTTPS, especifique um [uri](#) que contenha um nome de servidor seguro, como `https://example.com:443/test`. Para ver um tutorial completo sobre como criar uma API REST com uma integração privada, consulte [the section called “Tutorial: Crie uma API REST com uma integração privada”](#).

Criar uma integração privada

O procedimento a seguir mostra como criar uma integração privada que se conecte a um balanceador de carga utilizando um link da VPC V2.

Console de gerenciamento da AWS

Para ver um tutorial sobre como criar uma integração privada, consulte [the section called “Tutorial: Crie uma API REST com uma integração privada”](#).

AWS CLI

O comando [put-integration](#) indicado abaixo cria uma integração privada que se conecta a um balanceador de carga utilizando um link da 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
```

Em vez de fornecer diretamente o ID da conexão, você pode usar uma variável de estágio. Ao implantar a API em um estágio, você define o ID do link da VPC V2. O comando [put-integration](#) a seguir cria uma integração privada utilizando uma variável de estágio para o ID do link da 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}"
```

Inclua aspas duplas na expressão da variável de estágio (`\${stageVariables.vpcLinkV2Id}`) e insira um caractere de escape para \$.

OpenAPI

Você pode configurar uma API com a integração privada importando o arquivo OpenAPI da API. As configurações são semelhantes às definições do OpenAPI de uma API com integrações de HTTP, com as seguintes exceções:

- Você deve definir explicitamente `connectionType` como `VPC_LINK`.
- Você deve definir explicitamente `connectionId` como o ID de um `VpcLinkV2` ou como uma variável de estágio que faça referência ao ID de um `VpcLinkV2`.
- O parâmetro `uri` na integração privada aponta para um endpoint de HTTP/HTTPS na VPC, mas é usado para configurar o cabeçalho `Host` da solicitação de integração.
- O parâmetro `uri` na integração privada com um endpoint de HTTPS na VPC é usado para verificar o nome de domínio mencionado em relação ao nome no certificado instalado no VPC endpoint.

Você pode usar uma variável de estágio para fazer referência ao ID do VpcLinkV2. Ou você pode atribuir o valor do ID diretamente ao `connectionId`.

O arquivo OpenAPI em formato JSON a seguir mostra um exemplo de API com um link de VPC referenciado por uma variável de estágio (`${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.vpcLinkIdV2}",
          "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"
    }
  }
}
```

Atualizar uma integração privada

O exemplo a seguir atualiza o link da VPC V2 para uma integração privada.

Console de gerenciamento da AWS

Como atualizar uma integração privada

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST com uma integração privada.
3. Escolha o recurso e o método que usa uma integração privada.
4. Na guia Solicitação de integração em Configurações de solicitação de integração, selecione Editar.
5. Você pode editar a configuração da sua integração privada. Se você estiver usando atualmente um link da VPC V1, poderá alterá-lo para um link da VPC V2.
6. Escolha Salvar.
7. Implante a API novamente para que as alterações entrem em vigor.

AWS CLI

O seguinte comando [update-integration](#) atualiza uma integração privada para usar um link da 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\\"}]"]"
```

Integração privada utilizando links da VPC V1 (legado)

Note

A implementação de integrações privadas a seguir utiliza links da VPC V1. Os links da VPC V1 são recursos legados. Recomendamos que você utilize [links da VPC V2 para APIs REST](#).

Para criar uma integração privada, primeiro crie um link do Network Load Balancer. Seu Network Load Balancer deve ter um [listener](#) que envia solicitações para recursos em sua VPC. Para melhorar a disponibilidade da sua API, certifique-se de que o seu Network Load Balancer encaminhe o tráfego para recursos em mais de uma zona de disponibilidade na Região da AWS. Em seguida, você cria um link de VPC usado para conectar sua API e seu Network Load Balancer. Depois de criar um link de VPC, você cria integrações privadas para rotear o tráfego de sua API para recursos na VPC pelo link de VPC e pelo Network Load Balancer. O Network Load Balancer e a API devem pertencer à mesma Conta da AWS.

Tópicos

- [Configurar um Network Load Balancer para integrações privadas do API Gateway \(legado\)](#)
- [Conceder permissões para o API Gateway criar um link de VPC \(legado\)](#)
- [Configurar uma API do API Gateway com integrações privadas usando a AWS CLI \(legado\)](#)
- [Contas do API Gateway usadas para integrações privadas \(legado\)](#)

Configurar um Network Load Balancer para integrações privadas do API Gateway (legado)

Note

A implementação de integrações privadas a seguir utiliza links da VPC V1. Os links da VPC V1 são recursos legados. Recomendamos que você utilize [links da VPC V2 para APIs REST](#).

O procedimento a seguir descreve as etapas para configurar um Network Load Balancer (NLB) para as integrações privadas do API Gateway usando o console do Amazon EC2 e fornece referências para a obtenção de instruções detalhadas em cada etapa.

Para cada VPC na qual você tem recursos, só é necessário configurar um NLB e um VPCLink. O NLB oferece suporte a vários [listeners](#) e [grupos de destino](#) por NLB. É possível configurar cada serviço como um listener específico no NLB e usar um único VPCLink para se conectar ao NLB. Ao criar a integração privada no API Gateway, você define cada serviço usando a porta específica que é atribuída para cada serviço. Para obter mais informações, consulte [the section called “Tutorial: Crie uma API REST com uma integração privada”](#). O Network Load Balancer e a API devem pertencer à mesma Conta da AWS.

Para criar um Network Load Balancer para a integração privada usando o console do API Gateway

1. Faça login no Console de gerenciamento da AWS e abra o console do Amazon EC2 em <https://console.aws.amazon.com/ec2/>.
2. Configure um servidor web em uma instância do Amazon EC2. Para ver um exemplo de configuração, consulte [Tutorial: Instalar um servidor web LAMP no Amazon Linux 2](#).
3. Crie um Network Load Balancer, registre a instância do EC2 com um grupo de destino e adicione o grupo de destino a um listener do Network Load Balancer. Para obter detalhes, siga as instruções em [Conceitos básicos de Network Load Balancers](#).
4. Após a criação do Network Load Balancer, faça o seguinte:
 - a. Anote o ARN do Network Load Balancer. Você precisará dele para criar um link de VPC no API Gateway para integrar a API aos recursos da VPC por trás do Network Load Balancer.
 - b. Desative a avaliação do grupo de segurança para o PrivateLink.
 - Para desativar a avaliação do grupo de segurança para o tráfego do PrivateLink usando o console, escolha a guia Segurança e Editar. Em Configurações de segurança, desmarque a opção Aplicar regras de entrada ao tráfego do PrivateLink.

- Use o comando [set-security-groups](#) indicado abaixo para desativar a avaliação do grupo de segurança referente ao tráfego do 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

Não adicione nenhuma dependência aos CIDRs do API Gateway, pois elas podem ser alteradas sem aviso prévio.

Conceder permissões para o API Gateway criar um link de VPC (legado)

Note

A implementação de integrações privadas a seguir utiliza links da VPC V1. Os links da VPC V1 são recursos legados. Recomendamos que você utilize [links da VPC V2 para APIs REST](#).

Para que possa criar e manter um link de VPC, você, ou um usuário em sua conta, deve ter as permissões para criar, excluir e visualizar as configurações de serviço do VPC endpoint, alterar as permissões de serviço do VPC endpoint e examinar os load balancers. Para conceder essas permissões, use as seguintes etapas.

Para conceder as permissões de criação, atualização e exclusão de um link da VPC

1. Crie uma política do IAM semelhante a esta:

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": "*"
    }
]
}

```

Se você quiser habilitar a marcação para o link de VPC, permita as operações de marcação. Para obter mais informações, consulte [the section called “Permitir operações de marcação”](#).

2. Crie ou escolha uma função do IAM e anexe a política anterior à função.
3. Atribua a função do IAM a si mesmo ou ao usuário em sua conta que está criando os links da VPC.

Configurar uma API do API Gateway com integrações privadas usando a AWS CLI (legado)

Note

A implementação de integrações privadas a seguir utiliza links da VPC V1. Os links da VPC V1 são recursos legados. Recomendamos que você utilize [links da VPC V2 para APIs REST](#).

O tutorial a seguir mostra como usar a AWS CLI para criar um link de VPC e uma integração privada. Os seguintes pré-requisitos se aplicam:

- É necessário ter um Network Load Balancer criado e configurado com a origem da VPC como o destino. Para obter mais informações, consulte [Configurar um Network Load Balancer para integrações privadas do API Gateway \(legado\)](#). Isso deve ser feito na mesma Conta da AWS que a API. É necessário ter o ARN do Network Load Balancer para criar o link da VPC.
- Para criar e gerenciar um VpcLink, é preciso ter permissões para criar um VpcLink na API. Não é necessário ter permissões para usar o VpcLink. Para obter mais informações, consulte [Conceder permissões para o API Gateway criar um link de VPC \(legado\)](#).

Para configurar uma API com a integração privada usando a AWS CLI

1. Use o comando [create-vpc-link](#) a seguir para criar um VpcLink para o Network Load Balancer 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
```

A saída desse comando confirma o recebimento da solicitação e mostra o status PENDING da VpcLink que está sendo criada.

```
{
  "status": "PENDING",
  "targetArns": [
    "arn:aws:elasticloadbalancing:us-east-2:123456789012:loadbalancer/net/my-
vpclink-test-nlb/1234567890abcdef"
  ],
  "id": "gim7c3",
```

```
"name": "my-test-vpc-link"  
}
```

Demora de 2 a 4 minutos para que o API Gateway conclua a criação do VpcLink. Quando a operação é concluída com êxito, o status é AVAILABLE. É possível verificar isso usando o comando [get-vpc-link](#) indicado abaixo:

```
aws apigateway get-vpc-link --vpc-link-id gim7c3
```

Se a operação falhar, você obterá o status FAILED e a statusMessage conterá uma mensagem de erro. Por exemplo, se você tentar criar um VpcLink com um Network Load Balancer que já esteja associado a um VPC endpoint, obterá a seguinte mensagem na propriedade statusMessage:

```
"NLB is already associated with another VPC Endpoint Service"
```

Depois que a VpcLink for criada com sucesso, você poderá criar uma API e integrá-la ao recurso da VPC por meio da VpcLink.

Anote o valor de id do VpcLink recém-criado. Neste exemplo de saída, é *gim7c3*. Você precisará dele para configurar a integração privada.

2. Use o comando [create-rest-api](#) indicado abaixo para criar um recurso [RestApi](#) do API Gateway:

```
aws apigateway create-rest-api --name 'My VPC Link Test'
```

Anote o valor de id da RestApi e o valor de rootResourceId da RestApi no resultado obtido. Você precisa desse valor para executar outras operações na API.

Em seguida, crie uma API com apenas um método GET no recurso raiz (/) e integre o método ao VpcLink.

3. Use o comando [put-method](#) indicado abaixo para criar o método GET /:

```
aws apigateway put-method \  
  --rest-api-id abcdef123 \  
  --resource-id skpp60rab7 \  
  --http-method GET \  
  --authorization-type "NONE"
```

Se você não usa a integração de proxy com o VpcLink, também deve configurar pelo menos uma resposta do método para o código de status 200. Use a integração de proxy aqui.

- Depois de criar o método GET /, você configura a integração. Em uma integração privada, use o parâmetro `connection-id` para fornecer o ID do VpcLink. É possível usar uma variável de estágio ou inserir diretamente o ID do VpcLink. O parâmetro `uri` não é usado para rotear solicitações para o endpoint, mas sim para configurar o cabeçalho do Host e para a validação do certificado.

Use the VPC link ID

Use o comando [put-integration](#) indicado abaixo para utilizar o ID do VpcLink diretamente na integração:

```
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

Use o comando [put-integration](#) a seguir para fazer referência ao ID do link de VPC por meio de uma variável de estágio. Ao implantar a API em um estágio, você define o ID do link da 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 $stage.VPC_LINK_ID
```



```
--connection-id "${stageVariables.vpcLinkId}"
```

Inclua aspas duplas na expressão da variável de estágio (`${stageVariables.vpcLinkId}`) e insira um caractere de escape para o caractere \$.

A qualquer momento, também é possível atualizar a integração para alterar o `connection-id`. Use o comando [update-integration](#) indicado abaixo para atualizar a integração:

```
aws apigateway update-integration \  
  --rest-api-id abcdef123 \  
  --resource-id skpp60rab7 \  
  --http-method GET \  
  --patch-operations ' [{"op": "replace", "path": "/  
connectionId", "value": "${stageVariables.vpcLinkId}"} ] '
```

Certifique-se de usar uma lista JSON transformada em string como o valor do parâmetro `patch-operations`.

Como você usou a integração de proxy privada, a API agora está pronta para implantação e execução de testes.

5. Se você usou a variável de estágio para definir o `connection-id`, precisará implantar a API para testá-la. Use o comando [create-deployment](#) indicado abaixo para implantar a API com uma variável de estágio:

```
aws apigateway create-deployment \  
  --rest-api-id abcdef123 \  
  --stage-name test \  
  --variables vpcLinkId=gim7c3
```

Para atualizar a variável de estágio com um ID do VpcLink diferente (por exemplo, *asf9d7*), use o comando [update-stage](#) indicado abaixo:

```
aws apigateway update-stage \  
  --rest-api-id abcdef123 \  
  --stage-name test \  
  --patch-operations op=replace,path='/variables/vpcLinkId',value='asf9d7'
```

Ao codificar a propriedade `connection-id` com o ID literal do VpcLink, não é necessário implantar a API para testá-la. Use o comando [test-invoke-method](#) para testar a API antes que ela seja implantada.

6. Use o seguinte comando para invocar a API:

```
curl -X GET https://abcdef123.execute-api.us-east-2.amazonaws.com/test
```

Também é possível inserir o URL de invocação da API em um navegador da web para visualizar o resultado.

Contas do API Gateway usadas para integrações privadas (legado)

Os seguintes IDs de conta do API Gateway específicos da região são adicionados automaticamente ao serviço de VPC endpoint como `AllowedPrincipals` quando você cria um VpcLink.

Região	ID da conta
us-east-1	392220576650
us-east-2	718770453195
us-west-1	968246515281
us-west-2	109351309407
ca-central-1	796887884028
eu-west-1	631144002099
eu-west-2	544388816663
eu-west-3	061510835048
eu-central-1	474240146802
eu-central-2	166639821150
eu-north-1	394634713161

Região	ID da conta
eu-south-1	753362059629
eu-south-2	359345898052
ap-northeast-1	969236854626
ap-northeast-2	020402002396
ap-northeast-3	360671645888
ap-southeast-1	195145609632
ap-southeast-2	798376113853
ap-southeast-3	652364314486
ap-southeast-4	849137399833
ap-south-1	507069717855
ap-south-2	644042651268
ap-east-1	174803364771
sa-east-1	287228555773
me-south-1	855739686837
me-central-1	614065512851

Simular integrações para APIs REST no API Gateway

O Amazon API Gateway oferece suporte a integrações simuladas para métodos de API. Esse recurso permite que os desenvolvedores de API gerem respostas de API do API Gateway diretamente, sem necessidade de um backend de integração. Como desenvolvedor de API, você pode usar esse recurso para desbloquear as equipes dependentes que precisam trabalhar com uma API antes que o desenvolvimento do projeto seja concluído. Você também pode usar esse recurso para provisionar uma página de destino para sua API, que pode fornecer uma visão geral

e a navegação para a sua API. Para obter um exemplo dessa página de aterrissagem, consulte a solicitação e a resposta de integração do método GET no recurso raiz da API, discutida no exemplo [Tutorial: Criar uma API REST importando um exemplo](#).

Como desenvolvedor de APIs, você decide como o API Gateway responde a uma solicitação de integração simulada. Para isso, você configura a solicitação de integração e a resposta de integração do método para associar uma resposta a um determinado código de status. Para que um método com a integração de simulação retorne uma resposta 200, configure o modelo de mapeamento do corpo da solicitação de integração para retornar o seguinte.

```
{"statusCode": 200}
```

Configure uma resposta de integração 200 para ter o seguinte modelo de mapeamento do corpo, por exemplo:

```
{
  "statusCode": 200,
  "message": "Go ahead without me."
}
```

De forma parecida, para que o método retorne, por exemplo, uma resposta de erro 500, defina o modelo de mapeamento do corpo da solicitação de integração para retornar o seguinte.

```
{"statusCode": 500}
```

Defina uma resposta de integração 500 com, por exemplo, o seguinte modelo de mapeamento:

```
{
  "statusCode": 500,
  "message": "The invoked method is not supported on the API resource."
}
```

Como alternativa, você pode fazer com que um método de integração simulada retorne a resposta de integração padrão sem definir o modelo de mapeamento de solicitação de integração. A resposta de integração padrão é aquela com um HTTP status regex (Regex de status HTTP) indefinido. Certifique-se de que os comportamentos de passagem apropriados estejam definidos.

 Note

As integrações simuladas não são destinadas a oferecer suporte a modelos de respostas grandes. Se eles forem necessários ao seu caso de uso, considere usar a integração do Lambda.

Com um modelo de mapeamento de solicitação de integração, você pode fazer com que a lógica do aplicativo decida qual resposta de integração simulada deve ser retornada com base em certas condições. Por exemplo, você pode usar um parâmetro de consulta `scope` na solicitação recebida para determinar se será retornada uma resposta bem-sucedida ou uma resposta de erro:

```
{
  #if( $input.params('scope') == "internal" )
    "statusCode": 200
  #else
    "statusCode": 500
  #end
}
```

Dessa forma, o método de integração simulada permite a passagem das chamadas internas, ao mesmo tempo, rejeitando outros tipos de chamadas com uma resposta de erro.

Nesta seção, descrevemos como usar o console do API Gateway para habilitar a integração simulada para um método da API.

Tópicos

- [Ativar a integração simulada usando o console do API Gateway](#)

Ativar a integração simulada usando o console do API Gateway

É necessário ter o método disponível no API Gateway. Siga as instruções em [Tutorial: Crie uma API REST com uma integração de não proxy de HTTP](#).

1. Selecione um recurso de API e Crie método.

Para configurar o método, faça o seguinte:

- a. Em Tipo de método, selecione um método.

- b. Em Tipo de integração, selecione Simulação.
 - c. Escolha Criar método.
 - d. Na guia Solicitação de método, em Configurações de solicitação de método, selecione Editar.
 - e. Selecione Parâmetros de string de consulta de URL. Selecione Adicionar string de consulta e, em Nome, insira **scope**. Esse parâmetro de consulta determina se o autor da chamada é interno ou não.
 - f. Escolha Salvar.
2. Na guia Resposta do método, selecione Criar resposta e faça o seguinte:
 - a. Em Status HTTP, insira **500**.
 - b. Escolha Salvar.
3. Na guia Solicitação de integração, em Configurações de solicitação de integração, selecione Editar.
4. Selecione Modelos de mapeamento e, depois, faça o seguinte:
 - a. Escolha Add mapping template (Adicionar modelo de mapeamento).
 - b. Em Tipo de conteúdo, insira **application/json**.
 - c. Em Corpo do modelo, insira o seguinte:

```
{
  #if( $input.params('scope') == "internal" )
    "statusCode": 200
  #else
    "statusCode": 500
  #end
}
```
 - d. Escolha Salvar.
5. Na guia Resposta de integração, em Padrão - Resposta, selecione Editar.
6. Selecione Modelos de mapeamento e, depois, faça o seguinte:
 - a. Em Tipo de conteúdo, insira **application/json**.
 - b. Em Corpo do modelo, insira o seguinte:

```
{
```

```
"statusCode": 200,  
"message": "Go ahead without me"  
}
```

c. Escolha Salvar.

7. Selecione Criar resposta.

Para criar uma resposta 500, faça o seguinte:

- Em HTTP status regex (Regex de status HTTP), insira **5\d{2}**.
- Em Status de resposta do método, selecione **500**.
- Escolha Salvar.
- Em 5\d{2} - Resposta, selecione Editar.
- Selecione Modelos de mapeamento e, depois, Adicionar modelo de mapeamento.
- Em Tipo de conteúdo, insira **application/json**.
- Em Corpo do modelo, insira o seguinte:

```
{  
  "statusCode": 500,  
  "message": "The invoked method is not supported on the API resource."  
}
```

h. Escolha Salvar.

8. Selecione a guia Testar. Talvez seja necessário selecionar o botão de seta para a direita para mostrar a guia. Para testar a integração simulada, faça o seguinte:

- Digite `scope=internal` em Strings de consulta. Escolha Test (Testar). O resultado do teste mostra:

```
Request: /?scope=internal  
Status: 200  
Latency: 26 ms  
Response Body  
  
{  
  "statusCode": 200,  
  "message": "Go ahead without me"  
}
```

Response Headers

```
{"Content-Type":"application/json"}
```

- b. Digite `scope=public` em `Query strings` ou deixe em branco. Escolha `Test (Testar)`. O resultado do teste mostra:

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"}
```

Você também pode retornar os cabeçalhos em uma resposta de integração simulada, adicionando primeiro um cabeçalho à resposta de método e, em seguida, configurando um mapeamento de cabeçalho na resposta de integração. Na verdade, é assim que o console do API Gateway permite o suporte a CORS ao retornar os cabeçalhos necessários do CORS.

Solicitar validação para APIs REST no API Gateway

Você pode configurar o API Gateway para realizar a validação básica de uma solicitação de API antes de prosseguir com a solicitação de integração. Quando a validação falha, o API Gateway marca a solicitação como falha imediatamente, retorna uma resposta de erro 400 para o autor da chamada e publica os resultados da validação em CloudWatch Logs. Isso reduz as chamadas desnecessárias para o backend. O mais importante, isso permite concentrar-se nos esforços de validação específicos para o seu aplicativo. É possível validar um corpo de solicitação verificando se os parâmetros de solicitação necessários são válidos e não nulos especificando um esquema de modelo para uma validação de dados mais complicada.

Tópicos

- [Visão geral da validação básica de solicitações no API Gateway](#)
- [Modelos de dados para APIs REST](#)
- [Configurar a validação básica de solicitações no API Gateway](#)
- [Modelo do AWS CloudFormation de uma API de amostra com validação de solicitação básica](#)

Visão geral da validação básica de solicitações no API Gateway

O API Gateway pode realizar a validação básica da solicitação, para que você possa enfocar a validação específica do aplicativo no back-end. Para a validação, o API Gateway verifica uma das seguintes condições ou ambas:

- Se os parâmetros de solicitação necessários no URI, a string de consulta e os cabeçalhos de uma solicitação de entrada estão incluídos e não estão vazios. O API Gateway verifica apenas a existência de um parâmetro e não verifica o tipo ou o formato.
- Se a carga útil da solicitação aplicável atende à solicitação do [esquema JSON](#) do método para determinado tipo de conteúdo. Se nenhum tipo de conteúdo correspondente for encontrado, a validação da solicitação não será executada. Para usar o mesmo modelo, independentemente do tipo de conteúdo, defina o tipo de conteúdo do modelo de dados como `$default`.

Para ativar a validação, você deve especificar regras de validação em um [validador de solicitação](#), adicionar o validador ao [mapa de validadores de solicitação](#) da API e atribuir o validador a métodos de API individuais.

Note

Solicite a validação do corpo e os [Comportamento de solicitação de método para cargas úteis sem modelos de mapeamento para APIs REST no API Gateway](#) são dois tópicos distintos. Quando uma carga útil de solicitação não tem nenhum esquema de modelo correspondente, você pode optar por transmitir ou bloquear a carga útil original. Para obter mais informações, consulte [Comportamento de solicitação de método para cargas úteis sem modelos de mapeamento para APIs REST no API Gateway](#).

Modelos de dados para APIs REST

No API Gateway, um modelo define a estrutura de dados de uma carga. No API Gateway, modelos são definidos usando o [esquema JSON rascunho 4](#). O objeto JSON a seguir é uma amostra de dados no exemplo da Pet Store.

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

Os dados contêm o `id`, o `type` e o `price` do animal de estimação. Um modelo desses dados possibilita que você:

- Use a validação básica da solicitação.
- Crie modelos de mapeamento para transformação de dados.
- Crie um tipo de dados definido pelo usuário (UDT) ao gerar um SDK.

```
1 {  
2 "$schema": "http://json-schema.org/draft-04/ ← 1  
  schema#",  
3 "title": "PetStoreModel", ← 2  
4 "type": "object",  
5 "required": [ "price", "type" ], ← 3  
6 "properties": {  
7   "id": {  
8     "type": "integer" ← 4  
9   },  
10  "type": {  
11    "type": "string",  
12    "enum": [ "dog", "cat", "fish" ] ← 5  
13  },  
14  "price": { ← 6  
15    "type": "number",  
16    "minimum": 25.0,  
17    "maximum": 500.0  
18  }  
19 }  
20 }
```

The diagram illustrates a JSON Schema for a PetStoreModel. It consists of a JSON object with the following properties: `$schema`, `title`, `type`, `required`, and `properties`. The `properties` object contains three sub-objects: `id`, `type`, and `price`. The `id` object has a `type` property of `integer`. The `type` object has a `type` property of `string` and an `enum` property with values `dog`, `cat`, and `fish`. The `price` object has a `type` property of `number`, a `minimum` property of `25.0`, and a `maximum` property of `500.0`. Numbered annotations (1-6) point to specific parts of the schema: 1 points to the `$schema` property, 2 points to the `title` property, 3 points to the `required` array, 4 points to the `type` property of the `id` object, 5 points to the `enum` array of the `type` object, and 6 points to the `price` object.

Neste modelo:

1. O objeto `$schema` representa um identificador de versão do Esquema JSON válido. Esse esquema é o rascunho do JSON Schema v4.
2. O objeto `title` é um identificador humanamente legível para o modelo. Esse título é `PetStoreModel`.
3. A palavra-chave de validação `required` exige o `type` e o `price` para validação básica da solicitação.
4. As `properties` do modelo são `id`, `type` e `price`. Cada objeto tem propriedades que são descritas no modelo.

5. O objeto `type` pode ter somente os valores `dog`, `cat` ou `fish`.
6. O objeto `price` é um número e está limitado ao `minimum` de 25 e ao `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 }
```

Neste modelo:

1. Na linha 2, o objeto `$schema` representa um identificador de versão do JSON Schema válido. Esse esquema é o rascunho do JSON Schema v4.
2. Na linha 3, o objeto `title` é um identificador legível para o modelo. Esse título é `PetStoreModel`.
3. Na linha 5, a palavra-chave de validação `required` exige o `type` e o `price` para validação básica da solicitação.
4. Nas linhas 6 a 17, as `properties` do modelo são `id`, `type` e `price`. Cada objeto tem propriedades que são descritas no modelo.
5. Na linha 12, o objeto `type` pode ter somente os valores `dog`, `cat` ou `fish`.
6. Nas linhas 14 a 17, o objeto `price` é um número e está limitado ao `minimum` de 25 e ao `maximum` de 500.

Criar modelos mais complexos

É possível usar o `$ref` primitivo para criar definições reutilizáveis para modelos mais longos. Por exemplo, você pode criar uma definição chamada `Price` na seção `definitions` que descreve o objeto `price`. O valor de `$ref` é a definição `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
    }
  }
}
```

Você também pode referenciar outro esquema de modelo definido em um arquivo de modelo externo. Defina o valor da propriedade `$ref` para a localização do modelo. No exemplo a seguir, o modelo `Price` é definido no modelo `PetStorePrice` na API `a1234`.

```
{
  "$schema" : "http://json-schema.org/draft-04/schema#",
  "title" : "PetStorePrice",
  "type": "number",
  "minimum": 25,
  "maximum": 500
}
```

```
}
```

O modelo mais longo pode referenciar o 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"
    }
  }
}
```

Usar modelos de dados de saída

Se você transformar seus dados, poderá definir um modelo de carga útil na resposta de integração. Um modelo de carga útil pode ser usado quando você gera um SDK. Para linguagens de tipo forte, como Java, Objective-C ou Swift, o objeto corresponde a um tipo de dados definido pelo usuário (UDT). O API Gateway cria um UDT se você fornecê-lo com um modelo de dados ao gerar um SDK. Para ter mais informações sobre transformações de dados, consulte [Transformações de modelo de mapeamento para APIs REST no API Gateway](#).

O exemplo a seguir são dados de saída de uma resposta de integração.

```
{
  [
    {
      "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
  }
]
```

O exemplo a seguir é um modelo de payload que descreve os dados de saída.

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

Com esse modelo, você pode chamar um SDK para recuperar os valores de propriedades `description` e `askingPrice`, lendo as propriedades `PetStoreOutputModel[i].description` e `PetStoreOutputModel[i].askingPrice`. Se nenhum modelo for fornecido, o API Gateway usará o modelo vazio para criar um UDT padrão.

Próximas etapas

- Esta seção fornece recursos que você pode usar para ter mais conhecimento sobre os conceitos apresentados neste tópico.

É possível seguir os tutoriais de validação da solicitação:

- [Configurar a validação de solicitação usando o console do API Gateway](#)
- [Configurar a validação básica de solicitações usando a AWS CLI](#)

- [Configurar a validação básica de solicitações importando a definição de OpenAPI](#)
- Consulte mais informações sobre modelos de mapeamento e transformação de dados em [Transformações de modelo de mapeamento para APIs REST no API Gateway](#).

Configurar a validação básica de solicitações no API Gateway

Esta seção mostra como configurar a validação de solicitações para o API Gateway usando o console, a AWS CLI e uma definição de OpenAPI.

Tópicos

- [Configurar a validação de solicitação usando o console do API Gateway](#)
- [Configurar a validação básica de solicitações usando a AWS CLI](#)
- [Configurar a validação básica de solicitações importando a definição de OpenAPI](#)

Configurar a validação de solicitação usando o console do API Gateway

Você pode usar o console do API Gateway para validar uma solicitação selecionando um dos três validadores para uma solicitação de API:

- Validar o corpo.
- Valide parâmetros e cabeçalhos da string de consulta.
- Validar o corpo, parâmetros da string de consulta e cabeçalhos.

Ao aplicar um dos validadores em um método de API, o console do API Gateway adiciona o validador ao mapa [RequestValidators](#) da API.

Para seguir esse tutorial, você usará um modelo CloudFormation para criar uma API incompleta do API Gateway. Essa API tem um recurso `/validator` com os métodos GET e POST. Os dois métodos são integrados ao endpoint HTTP `http://petstore-demo-endpoint.execute-api.com/petstore/pets`. Você vai configurar dois tipos de validação de solicitação:

- No método GET, você vai configurar a validação da solicitação para os parâmetros da string de consulta de URL.
- No método POST, você vai configurar a validação da solicitação para o corpo da solicitação.

Isso vai possibilitar que somente determinadas chamadas de API sejam transmitidas à API.

Baixe e descompacte [o modelo de criação de aplicativos para CloudFormation](#). Você usará esse modelo para criar uma API incompleta. Você concluirá o restante das etapas no console do API Gateway.

Como criar uma pilha do CloudFormation

1. Abra o console do CloudFormation em <https://console.aws.amazon.com/cloudformation>.
2. Selecione Create stack (Criar pilha) e With new resources (standard) (Com novos recursos (padrão)).
3. Em Specify template (Especificar modelo), escolha Upload a template file (Fazer upload de um arquivo de modelo).
4. Selecione o modelo que você baixou.
5. Escolha Next (Próximo).
6. Em Nome da pilha, insira **request-validation-tutorial-console** e escolha Avançar.
7. Para Configurar opções de pilha, escolha Avançar.
8. Para Capabilities (Recursos), reconheça que CloudFormation pode criar recursos do IAM em sua conta.
9. Escolha Próximo e, depois, Enviar.

CloudFormationO fornece os recursos especificados no modelo. Pode demorar alguns minutos para concluir o provisionamento de seus recursos. Quando o status da sua pilha do CloudFormation for CREATE_COMPLETE, você estará pronto para passar para a próxima etapa.

Como selecionar sua API recém-criada

1. Selecione a pilha **request-validation-tutorial-console** recém-criada.
2. Escolha atributos.
3. Em ID físico, escolha sua API. Esse link direcionará você para o console do API Gateway.

Antes de modificar os métodos GET e POST, é necessário criar um modelo.

Como criar um modelo

1. É necessário um modelo para usar a validação da solicitação no corpo de uma solicitação recebida. Para criar um modelo, no painel de navegação principal, selecione Modelos.

2. Escolha Criar modelo.
3. Em Nome, digite **PetStoreModel**.
4. Em Tipo de conteúdo, insira **application/json**. Se nenhum tipo de conteúdo correspondente for encontrado, a validação da solicitação não será executada. Para usar o mesmo modelo, independentemente do tipo de conteúdo, insira **\$default**.
5. Em Descrição, insira **My PetStore Model** como descrição do modelo.
6. Em Esquema do modelo, cole o modelo a seguir no editor de código e escolha Criar.

```
{
  "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 obter mais informações sobre o modelo, consulte [Modelos de dados para APIs REST](#).

Como configurar a validação da solicitação para um método **GET**

1. No painel de navegação, selecione Recursos e, depois, selecione o método GET.
2. Na guia Solicitação de método, em Configurações de solicitação de método, escolha Editar.
3. Em Validador de solicitação, selecione Validar parâmetros de string de consulta e cabeçalhos.
4. Em Parâmetros de string de consulta de URL, faça o seguinte:

- a. Escolha Add query string (Adicionar string de consulta).
 - b. Em Nome, digite **petType**.
 - c. Ative a opção Obrigatório.
 - d. Mantenha Armazenamento em cache desativado.
5. Escolha Salvar.
6. Na guia Solicitação de integração, em Configurações de solicitação de integração, selecione Editar.
7. Em Parâmetros de string de consulta de URL, faça o seguinte:
- a. Escolha Add query string (Adicionar string de consulta).
 - b. Em Nome, digite **petType**.
 - c. Em Mapeado de, insira **method.request.querystring.petType**. Isso associa o **petType** ao tipo do animal de estimação.
- Para ter mais informações sobre mapeamento de dados, consulte [o tutorial de mapeamento de dados](#).
- d. Mantenha Armazenamento em cache desativado.
8. Escolha Salvar.

Como testar a validação da solicitação para o método **GET**

1. Selecione a guia Testar. Talvez seja necessário selecionar o botão de seta para a direita para mostrar a guia.
2. Em Strings de consulta, insira **petType=dog** e escolha Testar.
3. O teste de método vai gerar 200 OK e uma lista de cães.

Para ter informações sobre como transformar esses dados de saída, consulte o [tutorial de mapeamento de dados](#).

4. Remova **petType=dog** e escolha Testar.
5. O teste de método vai gerar um erro 400 com a seguinte mensagem de erro:

```
{
  "message": "Missing required request parameters: [petType]"
}
```

Como configurar a validação da solicitação para o método **POST**

1. No painel de navegação principal, selecione Recursos e, depois, selecione o método POST.
2. Na guia Solicitação de método, em Configurações de solicitação de método, escolha Editar.
3. Em Validador de solicitação, selecione Validar corpo.
4. Em Corpo da solicitação, escolha Adicionar modelo.
5. Em Tipo de conteúdo, insira **application/json**. Se nenhum tipo de conteúdo correspondente for encontrado, a validação da solicitação não será executada. Para usar o mesmo modelo, independentemente do tipo de conteúdo, insira `$default`.

Em Modelo, selecione PetStoreModel.

6. Escolha Salvar.

Como testar a validação da solicitação para um método **POST**

1. Selecione a guia Testar. Talvez seja necessário selecionar o botão de seta para a direita para mostrar a guia.
2. Em Corpo da solicitação, cole o seguinte no editor de código:

```
{
  "id": 2,
  "name": "Bella",
  "type": "dog",
  "price": 400
}
```

Escolha Testar.

3. O teste de método vai gerar `200 OK` e uma mensagem de êxito.
4. Em Corpo da solicitação, cole o seguinte no editor de código:

```
{
  "id": 2,
  "name": "Bella",
  "type": "dog",
  "price": 4000
}
```

Escolha Testar.

5. O teste de método vai gerar um erro 400 com a seguinte mensagem de erro:

```
{
  "message": "Invalid request body"
}
```

Na parte inferior dos logs de teste, o motivo do corpo de solicitação inválido será exibido. Nesse caso, o preço do animal de estimação estava fora do máximo especificado no modelo.

Para excluir uma pilha do CloudFormation

1. Abra o console do CloudFormation em <https://console.aws.amazon.com/cloudformation>.
2. Selecione sua pilha do CloudFormation.
3. Escolha Excluir e, em seguida, confirme sua escolha.

Próximas etapas

- Para ter informações sobre como transformar os dados de saída e realizar o mapeamento de mais dados, consulte o [tutorial de mapeamento de dados](#).
- Siga o tutorial [Configurar a validação básica da solicitação usando o AWS CLI](#), para realizar etapas semelhantes usando a AWS CLI.

Configurar a validação básica de solicitações usando a AWS CLI

Você pode criar um validador para configurar a validação da solicitação usando a AWS CLI. Para seguir esse tutorial, você usará um modelo CloudFormation para criar uma API incompleta do API Gateway.

 Note

Esse não é o mesmo modelo CloudFormation do tutorial do console.

Usando um recurso `/validator` pré-exposto, você criará os métodos GET e POST. Os dois métodos serão integrados ao endpoint HTTP `http://petstore-demo-endpoint.execute-api.com/petstore/pets`. Você vai configurar as duas validações de solicitação a seguir:

- No método GET, você criará um validador `params-only` para validar os parâmetros da string de consulta de URL.
- No método POST, você criará um validador `body-only` para validar o corpo da solicitação.

Isso vai possibilitar que somente determinadas chamadas de API sejam transmitidas à API.

Como criar uma pilha do CloudFormation

Baixe e descompacte [o modelo de criação de aplicativos para CloudFormation](#).

Para concluir o tutorial a seguir, é necessária a [AWS Command Line Interface \(AWS CLI\) versão 2](#).

Para comandos longos, um caractere de escape (`\`) é usado para dividir um comando em várias linhas.

Note

No Windows, alguns comandos da CLI do Bash que você costuma usar (como `zip`) não são compatíveis com os terminais integrados do sistema operacional. Para obter uma versão do Ubuntu com o Bash integrada no Windows, [instale o Subsistema do Windows para Linux](#). Os exemplos de comandos da CLI neste guia usam a formatação Linux. Os comandos que incluem documentos JSON em linha deverão ser reformatados se você estiver usando a CLI do Windows.

1. Use o comando a seguir para criar a pilha de CloudFormation.

```
aws cloudformation create-stack --stack-name request-validation-tutorial-cli
--template-body file://request-validation-tutorial-cli.zip --capabilities
CAPABILITY_NAMED_IAM
```

2. CloudFormationO fornece os recursos especificados no modelo. Pode demorar alguns minutos para concluir o provisionamento de seus recursos. Use o comando a seguir para ver o status de sua pilha de CloudFormation.

```
aws cloudformation describe-stacks --stack-name request-validation-tutorial-cli
```

- Quando o status da sua pilha de CloudFormation for `StackStatus: "CREATE_COMPLETE"`, use o comando a seguir para recuperar valores de saída relevantes para etapas futuras.

```
aws cloudformation describe-stacks --stack-name request-validation-tutorial-cli  
--query "Stacks[*].Outputs[*].{OutputKey: OutputKey, OutputValue: OutputValue,  
Description: Description}"
```

Os valores de saída são os seguintes:

- `Apild`, que é o ID da API. Para este tutorial, o ID da API é `abc123`.
- `ResourceId`, que é a ID do recurso validador em que os métodos GET e POST são expostos. Para este tutorial, o ID do recurso é `efg456`.

Como criar os validadores de solicitação e importar um modelo

- É necessário um validador para usar a validação da solicitação com o AWS CLI. Use o comando a seguir para criar um validador que valide somente os parâmetros da solicitação.

```
aws apigateway create-request-validator --rest-api-id abc123 \  
--no-validate-request-body \  
--validate-request-parameters \  
--name params-only
```

Anote o ID do validador `params-only`.

- Use o comando a seguir para criar um validador que valide somente o corpo da solicitação.

```
aws apigateway create-request-validator --rest-api-id abc123 \  
--validate-request-body \  
--no-validate-request-parameters \  
--name body-only
```

Anote o ID do validador `body-only`.

- É necessário um modelo para usar a validação da solicitação no corpo de uma solicitação recebida. Use o comando a seguir para importar um 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}}}'
```

Se nenhum tipo de conteúdo correspondente for encontrado, a validação da solicitação não será executada. Para usar o mesmo modelo, independentemente do tipo de conteúdo, especifique `$default` como a chave.

Como criar os métodos **GET** e **POST**

1. Use o comando a seguir para adicionar o método HTTP GET ao recurso `/validate`. Esse comando cria o método GET, adiciona o validador `params-only` e define a string de consulta `petType` conforme necessário.

```
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 o comando a seguir para adicionar o método HTTP POST ao recurso `/validate`. Esse comando cria o método POST, adiciona o validador `body-only` e anexa o modelo ao validador exclusivo do corpo.

```
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 o comando a seguir para configurar a resposta 200 OK do método como GET `/validate`.

```
aws apigateway put-method-response --rest-api-id abc123 \
  --resource-id efg456 \
```



```
--http-method GET \  
--status-code 200
```

Use o comando a seguir para configurar a resposta 200 OK do método como POST /validate.

```
aws apigateway put-method-response --rest-api-id abc123 \  
--resource-id efg456 \  
--http-method POST \  
--status-code 200
```

3. Use o comando a seguir para configurar uma Integration com um endpoint HTTP especificado para o 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 o comando a seguir para configurar uma Integration com um endpoint HTTP especificado para o 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 o comando a seguir para configurar a resposta de integração do método GET /validation.

```
aws apigateway put-integration-response --rest-api-id abc123 \  
--resource-id efg456 \  
--http-method GET \  
--status-code 200 \  
--selection-pattern ""
```

Use o comando a seguir para configurar a resposta de integração do método POST / validation.

```
aws apigateway put-integration-response --rest-api-id abc123 \  
    --resource-id efg456 \  
    --http-method POST \  
    --status-code 200 \  
    --selection-pattern ""
```

Como testar a API

1. Para testar o método GET, que executará a validação da solicitação para as strings de consulta, use o seguinte comando:

```
aws apigateway test-invoke-method --rest-api-id abc123 \  
    --resource-id efg456 \  
    --http-method GET \  
    --path-with-query-string '/validate?petType=dog'
```

O resultado retornará 200 OK e uma lista de cães.

2. Use o comando a seguir para testar sem incluir a string de consulta petType.

```
aws apigateway test-invoke-method --rest-api-id abc123 \  
    --resource-id efg456 \  
    --http-method GET
```

O resultado retornará um erro 400.

3. Para testar o método POST, que executará a validação da solicitação para o corpo da string, use o seguinte 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 }'
```

O resultado retornará 200 OK e uma mensagem de êxito.

4. Use o comando a seguir para testar usando um corpo inválido.

```
aws apigateway test-invoke-method --rest-api-id abc123 \  
    --resource-id efg456 \  
    --http-method POST \  
    --body '{"id": 1, "name": "bella", "type": "dog", "price" : 1000 }'
```

O resultado retornará um erro 400, pois o preço do cachorro está acima do preço máximo definido pelo modelo.

Para excluir uma pilha do CloudFormation

- Use o comando a seguir para excluir seus recursos de CloudFormation.

```
aws cloudformation delete-stack --stack-name request-validation-tutorial-cli
```

Configurar a validação básica de solicitações importando a definição de OpenAPI

Você pode declarar um validador de solicitação no nível da API especificando um conjunto de objetos [Objeto x-amazon-apigateway-request-validators.requestValidator](#) no mapa [Objeto x-amazon-apigateway-request-validators](#) para selecionar qual parte da solicitação será validada. No exemplo de definição de OpenAPI, há dois validadores:

- O validador `all` que valida o corpo, usando o modelo de dados `RequestBodyModel`, e os parâmetros.

O modelo de dados `RequestBodyModel` requer que o objeto JSON de entrada contenha as propriedades `name`, `type` e `price`. A propriedade `name` pode ser qualquer string, `type` deve ser um dos campos de enumeração especificados (`"dog"`, `"cat"`, `"fish"`) e `price` deve variar entre 25 e 500. O parâmetro `id` não é necessário.

- `param-only` que valida somente os parâmetros.

Para ativar um validador de solicitação em todos os métodos de uma API, especifique uma propriedade [Propriedade x-amazon-apigateway-request-validator](#) em nível de API da definição de OpenAPI. Na definição de OpenAPI de exemplo, o validador `all` é usado em todos os métodos de API, a menos que ele seja substituído de outra forma. Ao usar um modelo para validar o corpo, se nenhum tipo de conteúdo correspondente for encontrado, a validação da solicitação não será

executada. Para usar o mesmo modelo, independentemente do tipo de conteúdo, especifique `$default` como a chave.

Para ativar um validador de solicitação em um método individual, especifique a propriedade `x-amazon-apigateway-request-validator` no nível do método. No exemplo, definição de OpenAPI, o validador `param-only` sobreescreve o validador `all` no método GET.

Para importar o exemplo de OpenAPI para o API Gateway, consulte as instruções a seguir para [Importar uma API regional para o API Gateway](#) ou [Importar uma API otimizada para bordas para o 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" : {
          "$ref" : "#/definitions/ArrayOfError"
        },
        "headers" : {
          "test-method-response-header" : {
            "type" : "string"
          }
        }
      },
      "400" : {
        "description" : "400 response",
        "schema" : {
          "$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  
  }  
}
```

Modelo do AWS CloudFormation de uma API de amostra com validação de solicitação básica

A definição a seguir do modelo de exemplo do CloudFormation especifica uma API de amostra com a validação de solicitações habilitada. A API é um subconjunto da [API PetStore](#). Ela expõe um método POST para adicionar um animal de estimação à coleção pets e um método GET para consulta animais de estimação por um tipo especificado.

Há dois validadores de solicitações declarados:

GETValidator

Esse validador está habilitado no método GET. Ele permite que o API Gateway verifique se o parâmetro de consulta necessário (q1) está incluído e não está em branco na solicitação de entrada.

POSTValidator

Esse validador está habilitado no método POST. Isso permite que o API Gateway verifique se o formato da solicitação de carga está de acordo com o `RequestBodyModel` especificado quando o tipo de conteúdo é `application/json`. Caso nenhum tipo de conteúdo correspondente seja encontrado, a validação da solicitação não será realizada. Para usar o mesmo modelo, independentemente do tipo de conteúdo, especifique `$default`. `RequestBodyModel` contém um modelo adicional, `RequestBodyModelId`, para definir a identificação do animal de estimação.

```
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}'

```

Transformações de dados para APIs REST no API Gateway

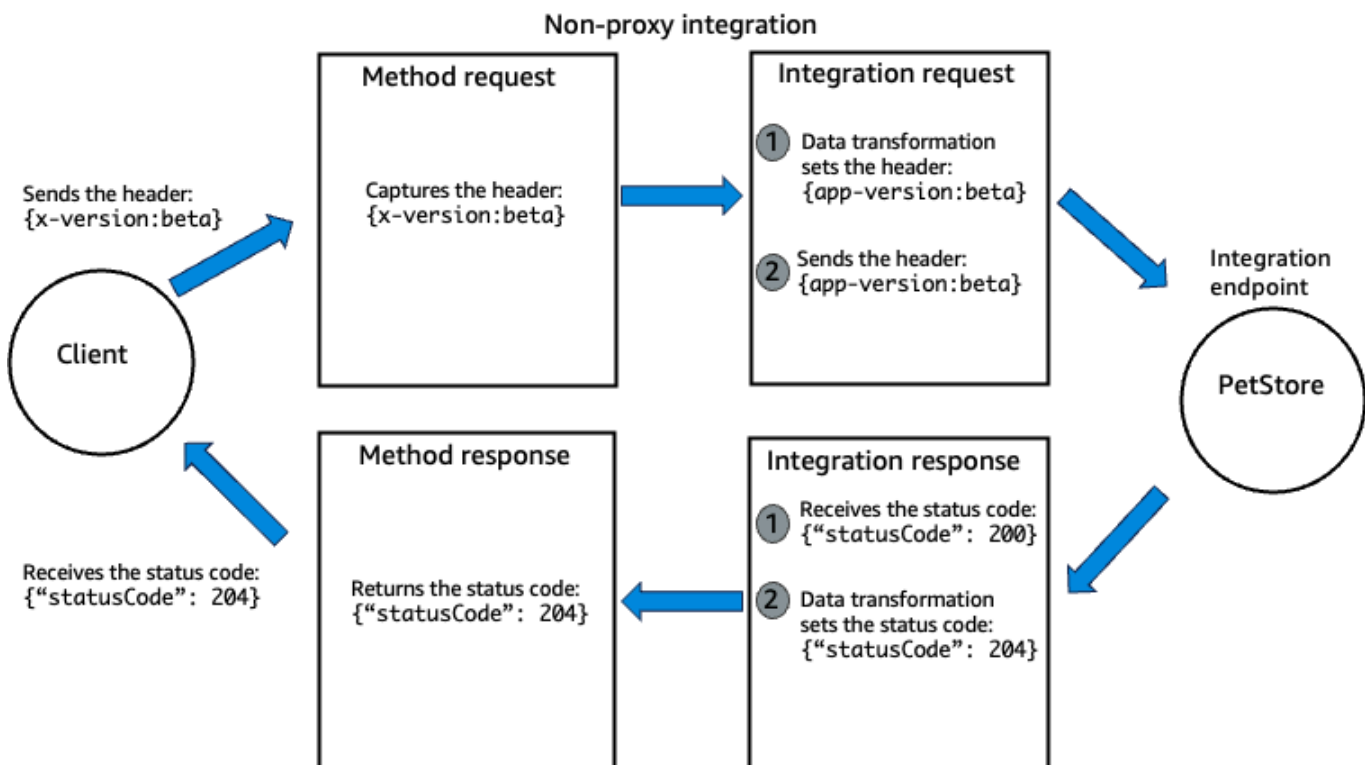
Note

Essa seção explica os recursos que você usa com uma integração sem proxy. No entanto, recomendamos que, quando possível, você use uma integração com proxy para a API REST. Uma integração com proxy HTTP tem uma configuração de integração simplificada e pode evoluir com o backend sem a necessidade de descartar a configuração existente. Para obter mais informações, consulte [the section called “Escolher um tipo de integração da API”](#).

Se usar uma integração sem proxy, você poderá utilizar dois recursos do API Gateway para transformar a solicitação de método e a resposta da integração. Você poderá transformar a solicitação do método se ela usar um formato de carga útil diferente da carga útil da solicitação de integração. Será possível transformar a resposta da integração se ela retornar um formato de

carga útil diferente do formato que você precisa retornar na resposta do método. Consulte mais informações sobre o ciclo de vida da solicitação em [the section called “Exemplo de recurso para uma API REST”](#).

O exemplo a seguir mostra uma transformação de dados em que, para o cabeçalho "x-version:beta", o parâmetro de cabeçalho x-version é transformado no parâmetro de cabeçalho app-version. A transformação dos dados de x-version para app-version ocorre na solicitação de integração. Dessa forma, o endpoint de integração recebe o valor do parâmetro do cabeçalho transformado. Quando o endpoint de integração retorna um código de status, esse código é transformado de 200 para 204 antes da resposta do método.



Para criar uma transformação de dados, é possível usar os seguintes recursos:

Mapeamento de parâmetros

No mapeamento de parâmetros, você pode modificar os parâmetros do caminho do URL da solicitação de integração, os parâmetros da string de consulta do URL ou os valores do cabeçalho HTTP, mas não pode modificar a carga útil da solicitação de integração. Você também pode modificar os valores do cabeçalho de resposta HTTP. Use o mapeamento de parâmetros para criar valores de cabeçalho estáticos para compartilhamento de recursos de origem cruzada (CORS).

É possível usar o mapeamento de parâmetros em sua solicitação de integração para integrações de proxy e não proxy, mas para usar o mapeamento de parâmetros para uma resposta de integração, você precisa de uma integração não proxy. O mapeamento de parâmetros não requer nenhuma programação em [Velocity Template Language \(VTL\)](#). Para obter mais informações, consulte [the section called “Mapeamento de parâmetros”](#).

Transformações de modelo de mapeamento

Nas transformações de modelo de mapeamento, você usa um modelo de mapeamento para mapear parâmetros de caminho de URL, parâmetros de string de consulta de URL, cabeçalhos HTTP e o corpo da solicitação de integração ou resposta de integração. Modelo de mapeamento é um script expresso em [Velocity Template Language \(VTL\)](#) que usa [expressões JSONPath](#) e é aplicado à carga útil com base no cabeçalho Content-type.

Com um modelo de mapeamento, é possível fazer o seguinte:

- Selecionar quais dados enviar usando a integração a Serviços da AWS, como funções do Amazon DynamoDB ou do Lambda ou endpoints HTTP. Para obter mais informações, consulte [the section called “Tutorial: modificar a solicitação e a resposta de integração para integrações a serviços da AWS”](#).
- Substituir condicionalmente os parâmetros de solicitação e resposta de integração de uma API, criar valores de cabeçalho e substituir códigos de status. Para obter mais informações, consulte [the section called “Substituir parâmetros de solicitação e resposta e códigos de status de API.”](#).

Você também pode especificar o comportamento da API quando um corpo de solicitação de integração tem um cabeçalho Content-type sem modelos de mapeamento correspondentes. Isso é chamado de comportamento de passagem direta de integração. Para obter mais informações, consulte [the section called “Comportamento de solicitação de método para cargas úteis sem modelos de mapeamento”](#).

Escolher entre mapeamento de parâmetros e transformações de modelos de mapeamento

Recomendamos que você use o mapeamento de parâmetros para transformar dados quando possível. Se a API exigir que você altere o corpo ou execute substituições e modificações condicionais com base na solicitação de integração recebida ou na resposta de integração, e não for possível usar uma integração com proxy, use transformações de modelo de mapeamento.

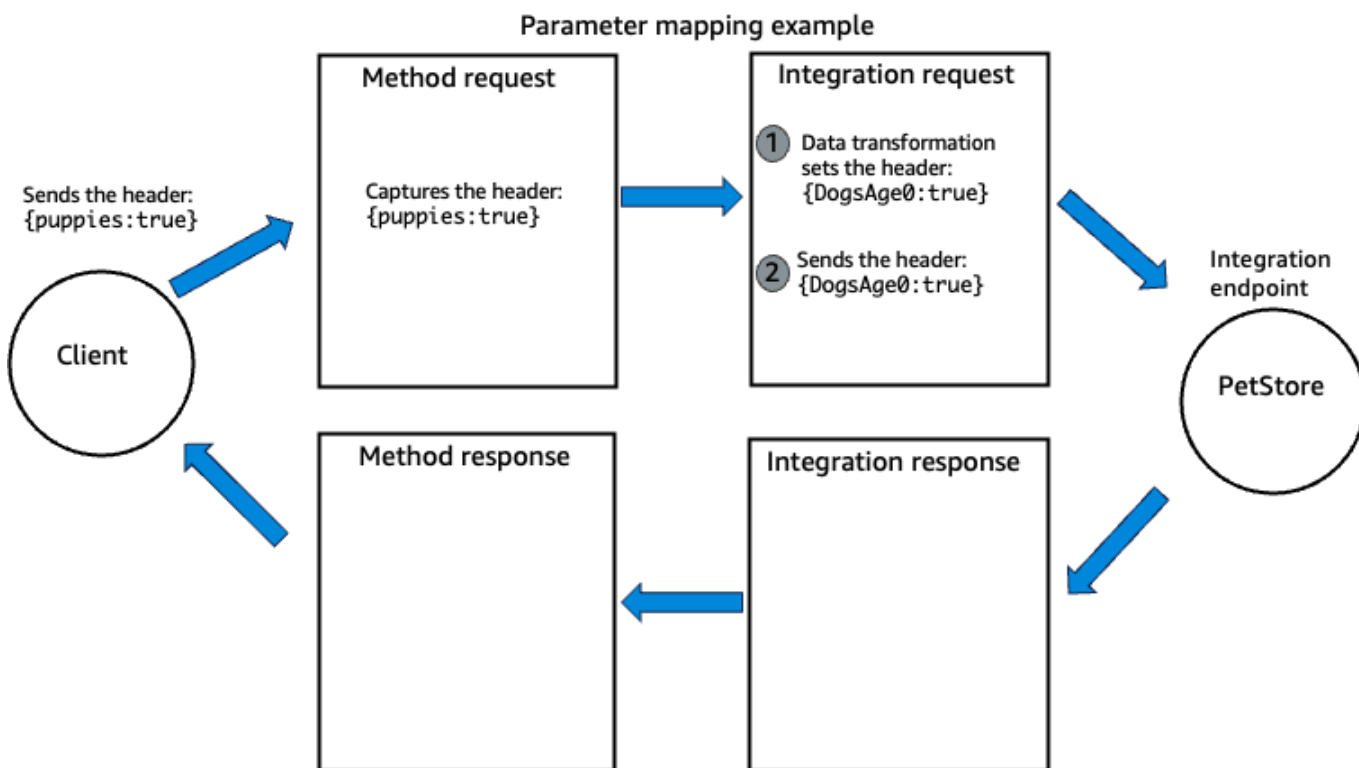
Mapeamento de parâmetros para APIs REST no API Gateway

Note

Se você estiver usando uma API HTTP, consulte [the section called “Mapeamento de parâmetros”](#).

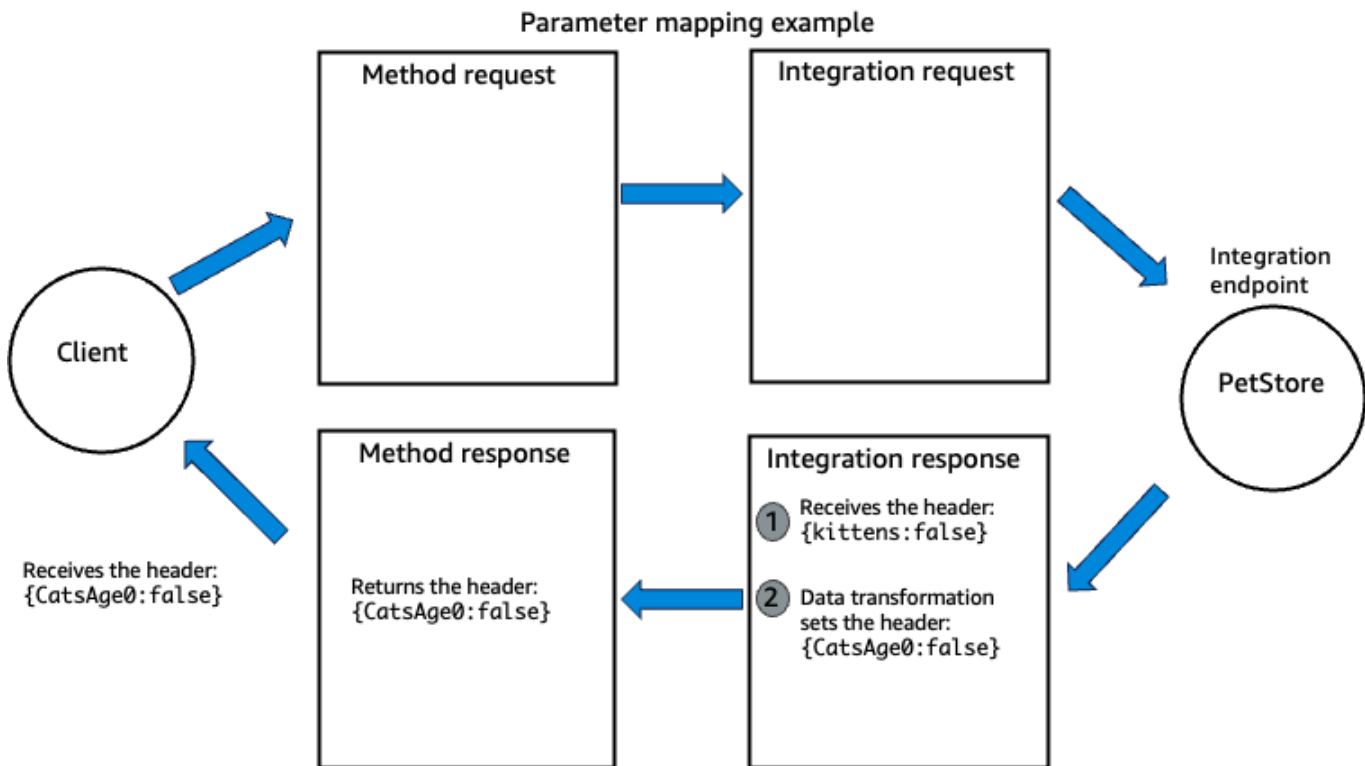
No mapeamento de parâmetros, você mapeia os parâmetros de solicitação ou resposta. É possível mapear parâmetros usando expressões de mapeamento de parâmetros ou valores estáticos. Consulte uma lista de expressões de mapeamento em [the section called “Referência da origem do mapeamento de parâmetros”](#). É possível usar o mapeamento de parâmetros em sua solicitação de integração para integrações de proxy e não proxy, mas para usar o mapeamento de parâmetros para uma resposta de integração, você precisa de uma integração não proxy.

Por exemplo, você pode mapear o parâmetro do cabeçalho da solicitação do método `puppies` para o parâmetro do cabeçalho da solicitação de integração `DogsAge0`. Então, se um cliente enviar o cabeçalho `puppies:true` para a API, a solicitação de integração enviará o cabeçalho da solicitação `DogsAge0:true` ao endpoint de integração. O diagrama a seguir mostra o ciclo de vida da solicitação nesse exemplo.



Para criar esse exemplo usando o API Gateway, consulte [the section called “Exemplo 1: mapear um parâmetro de solicitação de método para um parâmetro de solicitação de integração”](#).

Como outro exemplo, você também pode mapear o parâmetro do cabeçalho de resposta da integração kittens para o parâmetro do cabeçalho de resposta do método CatsAge0. Então, se o endpoint de integração retornar `kittens:false`, o cliente receberá o cabeçalho `CatsAge0:false`. O diagrama a seguir mostra o ciclo de vida da solicitação nesse exemplo.



Tópicos

- [Exemplos de mapeamento de parâmetros para APIs REST no API Gateway](#)
- [Referência da origem do mapeamento de parâmetros para APIs REST no API Gateway](#)

Exemplos de mapeamento de parâmetros para APIs REST no API Gateway

Os exemplos a seguir mostram como criar expressões de mapeamento de parâmetros usando o console do API Gateway, a OpenAPI e modelos do CloudFormation. Consulte um exemplo de como usar o mapeamento de parâmetros para criar os cabeçalhos CORS necessários em [the section called “CORS”](#).

Exemplo 1: mapear um parâmetro de solicitação de método para um parâmetro de solicitação de integração

O exemplo a seguir mapeia o parâmetro do cabeçalho da solicitação do método `puppies` para o parâmetro do cabeçalho da solicitação de integração `DogsAge0`.

Console de gerenciamento da AWS

Como mapear o parâmetro de solicitação do método

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST.
3. Escolha um método.

O método deve ter uma integração sem proxy.

4. Em Configurações de solicitação de método, selecione Editar.
5. Escolha Cabeçalhos de solicitação HTTP.
6. Escolha Add header (Adicionar cabeçalho).
7. Em Nome, digite **puppies**.
8. Escolha Salvar.
9. Selecione a guia Solicitação de integração e, em Configurações de solicitação de integração, selecione Editar.

O Console de gerenciamento da AWS adiciona automaticamente um mapeamento de parâmetros de `method.request.header.puppies` para `puppies` para você, mas é necessário alterar o Nome para corresponder ao parâmetro do cabeçalho da solicitação que é esperado pelo endpoint de integração.

10. Em Nome, digite **DogsAge0**.
11. Escolha Salvar.
12. Implante a API novamente para que as alterações entrem em vigor.

As etapas a seguir mostram como verificar se o mapeamento de parâmetros foi bem-sucedido.

(Opcional) Testar o mapeamento de parâmetros

1. Selecione a guia Testar. Talvez seja necessário selecionar o botão de seta para a direita para mostrar a guia.
2. Para os cabeçalhos, insira **puppies:true**.
3. Escolha Testar.
4. Em Logs, o resultado deve ser algo semelhante a:

```
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}
```

O parâmetro do cabeçalho da solicitação foi alterado de puppies para DogsAge0.

CloudFormation

Nesse exemplo, você usa a propriedade [corpo](#) para importar um arquivo de definição da OpenAPI para o 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"
    }
}
}
}
}
}

```

Exemplo 2: mapear vários parâmetros de solicitação de método para diferentes parâmetros de solicitação de integração

O exemplo a seguir mapeia o parâmetro de string de consulta de vários valores da solicitação do método `methodRequestQueryParam` para o parâmetro de string de consulta da solicitação de integração `integrationQueryParam` e mapeia o parâmetro de cabeçalho da solicitação do método `methodRequestHeaderParam` para o parâmetro de caminho da solicitação de integração `integrationPathParam`.

Console de gerenciamento da AWS

Como mapear os parâmetros da solicitação de método

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST.
3. Escolha um método.

O método deve ter uma integração sem proxy.

4. Em Configurações de solicitação de método, selecione Editar.
5. Selecione Parâmetros de string de consulta de URL.
6. Escolha Add query string (Adicionar string de consulta).
7. Em Nome, digite **methodRequestQueryParam**.
8. Escolha Cabeçalhos de solicitação HTTP.
9. Escolha Add header (Adicionar cabeçalho).
10. Em Nome, digite **methodRequestHeaderParam**.
11. Escolha Salvar.
12. Selecione a guia Solicitação de integração e, em Configurações de solicitação de integração, selecione Editar.
13. Selecione Parâmetros de caminho de URL.
14. Selecione Adicionar parâmetro de caminho.
15. Em Nome, digite **integrationPathParam**.
16. Em Mapeado de, insira **method.request.header.methodRequestHeaderParam**.

Isso mapeia o cabeçalho da solicitação de método que você especificou na solicitação de método para um novo parâmetro de caminho de solicitação de integração.

17. Selecione Parâmetros de string de consulta de URL.
18. Escolha Add query string (Adicionar string de consulta).
19. Em Nome, digite **integrationQueryParam**.
20. Em Mapeado de,
insira **method.request.multivaluequerystring.methodRequestQueryParam**.

Isso mapeia o parâmetro da string de consulta de vários valores para um novo parâmetro de string de consulta de solicitação de integração de valor único.

21. Escolha Salvar.

22. Implante a API novamente para que as alterações entrem em vigor.

CloudFormation

Nesse exemplo, você usa a propriedade [corpo](#) para importar um arquivo de definição da OpenAPI para o API Gateway.

A definição da OpenAPI a seguir cria os seguintes mapeamentos de parâmetros para uma integração HTTP:

- O cabeçalho da solicitação de método, chamado `methodRequestHeaderParam`, no parâmetro do caminho de solicitação de integração, chamado `integrationPathParam`
- A string de consulta da solicitação de método de vários valores, chamada `methodRequestQueryParam`, na string de consulta da solicitação de integração, chamada `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

A definição da OpenAPI a seguir cria os seguintes mapeamentos de parâmetros para uma integração HTTP:

- O cabeçalho da solicitação de método, chamado `methodRequestHeaderParam`, no parâmetro do caminho de solicitação de integração, chamado `integrationPathParam`
- A string de consulta da solicitação de método de vários valores, chamada `methodRequestQueryParam`, na string de consulta da solicitação de integração, chamada `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"
  }
}
```

Exemplo 3: mapear campos do corpo da solicitação JSON para os parâmetros da solicitação de integração

Também é possível mapear parâmetros da solicitação de integração com base em campos no corpo da solicitação JSON usando uma [expressão JSONPath](#). O exemplo a seguir mapeia o corpo da solicitação de método para um cabeçalho da solicitação de integração chamado `body-header` e mapeia parte do corpo da solicitação, conforme expresso por uma expressão JSON, para um cabeçalho da solicitação de integração chamado `pet-price`.

Para testar esse exemplo, forneça uma entrada que contenha uma categoria de preço, como a seguinte:

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

Console de gerenciamento da AWS

Como mapear os parâmetros da solicitação de método

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST.
3. Escolha um método POST, PUT, PATCH ou ANY.

O método deve ter uma integração sem proxy.

4. Em Configurações de solicitação de integração, selecione Editar.
5. Selecione Parâmetros de cabeçalhos de solicitações de URL.
6. Selecione Adicionar parâmetro de cabeçalho de solicitação.
7. Em Nome, digite **body-header**.
8. Em Mapeado de, insira **method.request.body**.

Isso mapeia o corpo da solicitação de método para um novo parâmetro de cabeçalho de solicitação de integração.

9. Selecione Adicionar parâmetro de cabeçalho de solicitação.
10. Em Nome, digite **pet-price**.
11. Em Mapeado de, insira **method.request.body[0].price**.

Isso mapeia uma parte do corpo da solicitação do método para um novo parâmetro de cabeçalho da solicitação de integração.

12. Escolha Salvar.
13. Implante a API novamente para que as alterações entrem em vigor.

CloudFormation

Nesse exemplo, você usa a propriedade [corpo](#) para importar um arquivo de definição da OpenAPI para o 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

A definição de OpenAPI a seguir mapeia parâmetros de solicitação de integração com base nos campos no corpo da solicitação 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"
        }
      }
    }
  }
}
```

Exemplo 4: mapear a resposta de integração para a resposta do método

Também é possível mapear a resposta de integração para a resposta do método. O exemplo a seguir mapeia o corpo da resposta de integração para um cabeçalho de resposta de método chamado `location`, mapeia o cabeçalho de resposta de integração `x-app-id` para o cabeçalho de resposta de método `id` e mapeia o cabeçalho de resposta de integração de vários valores `item` para o cabeçalho de resposta do método `items`.

Console de gerenciamento da AWS

Como mapear a resposta de integração

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST.
3. Escolha um método.

O método deve ter uma integração sem proxy.

4. Escolha a guia Resposta do método e, em Resposta 200, selecione Editar.
5. Em Nome do cabeçalho, escolha Adicionar cabeçalho.
6. Crie três cabeçalhos chamados **id**, **item** e **location**.
7. Escolha Salvar.
8. Escolha a guia Resposta de integração e em Padrão: resposta, selecione Editar.
9. Em Mapeamentos de cabeçalho, insira o indicado a seguir.
 - a. Em id, insira **integration.response.header.x-app-id**.
 - b. Em item, insira **integration.response.multivalueheader.item**
 - c. Em local, insira **integration.response.body.redirect.url**.
10. Escolha Save (Salvar).
11. Implante a API novamente para que as alterações entrem em vigor.

CloudFormation

Nesse exemplo, você usa a propriedade `corpo` para importar um arquivo de definição da OpenAPI para o 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

A definição de OpenAPI a seguir mapeia a resposta de integração para a resposta do 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
    }
}
}
}

```

Referência da origem do mapeamento de parâmetros para APIs REST no API Gateway

Ao criar um mapeamento de parâmetros, especifique os parâmetros de solicitação de método ou resposta de integração a serem modificados e especifique como modificar esses parâmetros.

A tabela a seguir mostra os parâmetros de solicitação de método que você pode mapear e a expressão para criar o mapeamento. Nessas expressões, *nome* é o nome de um parâmetro

de solicitação de método. Por exemplo, para mapear o parâmetro do cabeçalho da solicitação `puppies`, use a expressão `method.request.header.puppies`. A expressão deve corresponder à expressão regular `'^[a-zA-Z0-9._$-]+$'`. É possível usar o mapeamento de parâmetros em sua solicitação de integração para integrações de proxy e não proxy.

Fonte de dados mapeada	Expressão de mapeamento
Caminho de solicitação de método	<code>method.request.path.</code> <i>name</i>
String de consulta da solicitação de método	<code>method.request.querystring.</code> <i>name</i>
String de consulta de solicitação do método de vários valores	<code>method.request.multivaluequerystring.</code> <i>name</i>
Cabeçalho da solicitação de método	<code>method.request.header.</code> <i>name</i>
Cabeçalho de solicitação de método de vários valores	<code>method.request.multivalueheader.</code> <i>name</i>
Corpo de solicitação de método	<code>method.request.body</code>
Corpo de solicitação de método (JsonPath)	<code>method.request.body.</code> <i>JSONPath_EXPRESSION</i> . <i>JSONPath_EXPRESSION</i> é uma expressão JSONPath para um campo JSON do corpo de uma solicitação. Consulte mais informações em Expressão JSONPath .
Variáveis de estágio	<code>stageVariables.</code> <i>name</i>
Variáveis de contexto	<code>context.</code> <i>name</i> O nome deve ser uma das variáveis de contexto compatíveis .
Valor estático	<code>'static_value'</code> .

Fonte de dados mapeada	Expressão de mapeamento
	<i>static_value</i> é um literal de string e deve estar entre aspas simples. Por exemplo, 'https://www.example.com' .

A tabela a seguir mostra os parâmetros da resposta de integração que você pode mapear e a expressão para criar o mapeamento. Nessas expressões, *nome* é o nome de um parâmetro de resposta de integração. É possível mapear cabeçalhos de resposta do método por meio de qualquer cabeçalho de resposta de integração ou corpo de resposta de integração, variáveis \$context ou valores estáticos. Para usar o mapeamento de parâmetros para uma resposta de integração, você precisa de uma integração não proxy.

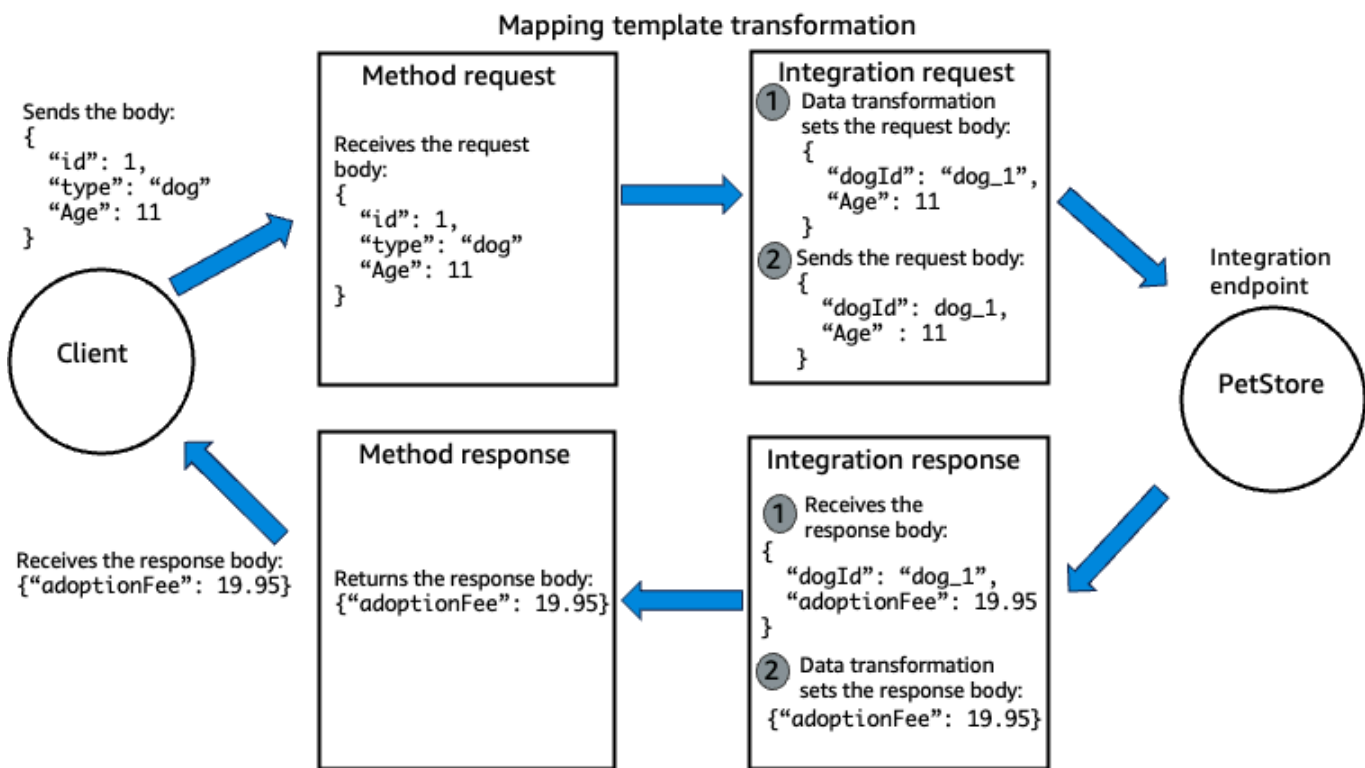
Fonte de dados mapeada	Expressão de mapeamento
Cabeçalho da resposta de integração	integration.response.header. <i>nome</i>
Cabeçalho da resposta de integração	integration.response.multivalueheader. <i>nome</i>
Corpo da resposta de integração	integration.response.body
Corpo da resposta de integração (JsonPath)	integration.response.body. <i>JSONPath_EXPRESSION</i> <i>JSONPath_EXPRESSION</i> é uma expressão JSONPath para um campo JSON do corpo de uma resposta. Consulte mais informações em Expressão JSONPath .
Variável de estágio	stageVariables. <i>nome</i>
Variável de contexto	context. <i>nome</i> O nome deve ser uma das variáveis de contexto compatíveis .
Valor estático	' <i>static_value</i> '

Fonte de dados mapeada	Expressão de mapeamento
	<i>static_value</i> é um literal de string e deve estar entre aspas simples. Por exemplo, 'https://www.example.com' .

Transformações de modelo de mapeamento para APIs REST no API Gateway

Uma transformação de modelo de mapeamento usa um modelo de mapeamento para modificar a solicitação ou a resposta de integração. Modelo de mapeamento é um script expresso em [Velocity Template Language \(VTL\)](#) e é aplicado a uma carga útil usando o [JSONPath](#) baseado no cabeçalho Content-type. Você utiliza modelos de mapeamento ao usar transformações de modelos de mapeamento. Esta seção descreve informações conceituais relacionadas aos modelos de mapeamento.

O diagrama a seguir mostra o ciclo de vida da solicitação para um recurso POST /pets que tem uma integração com um endpoint de integração da PetStore. Nessa API, um usuário envia dados sobre um animal de estimação e o endpoint de integração retorna a taxa de adoção associada a um animal de estimação. Nesse ciclo de vida da solicitação, as transformações do modelo de mapeamento filtram o corpo da solicitação para o endpoint de integração e filtram o corpo da resposta do endpoint de integração.



As seções a seguir explicam o ciclo de vida da solicitação e da resposta.

Solicitação de método e solicitação de integração

No exemplo anterior, se este for o corpo da solicitação enviado para a solicitação do método:

```

POST /pets
HTTP/1.1
Host:abcd1234.us-west-2.amazonaws.com
Content-type: application/json

{
  "id": 1,
  "type": "dog",
  "Age": 11,
}
```

Esse corpo da solicitação não está no formato correto para ser usado pelo endpoint de integração, então o API Gateway executa uma transformação do modelo de mapeamento. O API Gateway só executa uma transformação do modelo de mapeamento porque há um modelo de mapeamento definido para `application/json` de Content-Type. Se você não definir um modelo de

mapeamento para Content-Type, por padrão, o API Gateway passará o corpo por meio da solicitação de integração ao endpoint de integração. Para modificar esse comportamento, consulte [the section called “Comportamento de solicitação de método para cargas úteis sem modelos de mapeamento”](#).

O modelo de mapeamento a seguir transforma os dados da solicitação de método na solicitação de integração antes de serem enviados ao endpoint de integração:

```
#set($inputRoot = $input.path('$'))
{
  "dogId" : "dog_"$elem.id,
  "Age": $inputRoot.Age
}
```

1. A variável `$inputRoot` representa o objeto raiz nos dados JSON originais da seção anterior. As diretivas começam com o símbolo `#`.
2. `dog` é uma concatenação do `id` do usuário e de um valor de string.
3. `Age` é do corpo da solicitação do método.

Então, a saída a seguir é encaminhada ao endpoint de integração:

```
{
  "dogId" : "dog_1",
  "Age": 11
}
```

Resposta de integração e resposta do método

Após a solicitação bem-sucedida ao endpoint de integração, o endpoint envia uma resposta à resposta de integração do API Gateway. Veja a seguir exemplos de dados de saída do endpoint de integração:

```
{
  "dogId" : "dog_1",
  "adoptionFee": 19.95,
}
```

A resposta do método espera uma carga útil diferente da retornada pela resposta de integração. O API Gateway executa uma transformação do modelo de mapeamento. O API Gateway só executa uma transformação do modelo de mapeamento porque há um modelo de mapeamento definido

para `application/json` de `Content-Type`. Se você não definir um modelo de mapeamento para `Content-Type`, por padrão, o API Gateway passará o corpo por meio da resposta de integração para a resposta do método. Para modificar esse comportamento, consulte [the section called “Comportamento de solicitação de método para cargas úteis sem modelos de mapeamento”](#).

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

Veja a seguir a saída enviada à resposta do método:

```
{"adoptionFee": 19.95}
```

Isso conclui o exemplo de transformação do modelo de mapeamento. Recomendamos que, quando possível, em vez de usar transformações de modelo de mapeamento, você use uma integração com proxy para transformar dados. Para obter mais informações, consulte [the section called “Escolher um tipo de integração da API”](#).

Comportamento de solicitação de método para cargas úteis sem modelos de mapeamento para APIs REST no API Gateway

Se a solicitação de método tiver uma carga útil e você não tiver um modelo de mapeamento definido para o cabeçalho `Content-Type`, poderá optar por passar a carga útil da solicitação fornecida pelo cliente por meio da solicitação de integração ao backend sem transformação. O processo é conhecido como passagem direta de integração.

O comportamento real de passagem de uma solicitação recebida é determinado por essa configuração. Existem três opções:

Quando nenhum modelo corresponder ao cabeçalho `Content-Type` da solicitação

Escolha essa opção se desejar que o corpo da solicitação de método passe a solicitação de integração para o back-end sem transformação quando o tipo de conteúdo da solicitação de método não corresponder a nenhum dos tipos de conteúdo associados aos modelos de mapeamento.

Ao chamar a API do API Gateway, escolha essa opção definindo `WHEN_NO_MATCH` como o valor da propriedade `passthroughBehavior` na [Integração](#).

Quando não há modelos definidos (recomendado)

Escolha essa opção se desejar que o corpo da solicitação de método passe a solicitação de integração para o back-end sem transformação quando nenhum modelo de mapeamento estiver definido na solicitação de integração. Se um modelo for definido quando essa opção for selecionada, a solicitação de método de um tipo de conteúdo e carga útil que não corresponda a nenhum modelo de mapeamento definido será rejeitada com uma resposta HTTP 415 Tipo de mídia incompatível.

Ao chamar a API do API Gateway, escolha essa opção definindo `WHEN_NO_TEMPLATES` como o valor da propriedade `passthroughBehavior` na [Integração](#).

Nunca

Escolha essa opção se não desejar que o corpo da solicitação de método passe a solicitação de integração para o back-end sem transformação quando nenhum modelo de mapeamento estiver definido na solicitação de integração. Se um modelo for definido quando essa opção for selecionada, a solicitação de método de um tipo de conteúdo não mapeado será rejeitada com uma resposta HTTP 415 Tipo de mídia sem suporte.

Ao chamar a API do API Gateway, escolha essa opção definindo `NEVER` como o valor da propriedade `passthroughBehavior` na [Integração](#).

Os exemplos a seguir mostram os possíveis comportamentos de passagem.

Exemplo 1: um modelo de mapeamento é definido na solicitação de integração para o tipo de conteúdo `application/json`.

Content-type	Opção de passagem	Comportamento
Nenhum O padrão do API Gateway é <code>application/json</code>	<code>WHEN_NO_MATCH</code>	A carga da solicitação é transformada usando o modelo.
Nenhum O padrão do API Gateway é <code>application/json</code>	<code>WHEN_NO_TEMPLATES</code>	A carga da solicitação é transformada usando o modelo.

Content-type	Opção de passagem	Comportamento
Nenhum O padrão do API Gateway é application/json	NEVER	A carga da solicitação é transformada usando o modelo.
application/json	WHEN_NO_MATCH	A carga da solicitação é transformada usando o modelo.
application/json	WHEN_NO_TEMPLATES	A carga da solicitação é transformada usando o modelo.
application/json	NEVER	A carga da solicitação é transformada usando o modelo.
application/xml	WHEN_NO_MATCH	A carga da solicitação não é transformada e é enviada ao backend no estado em que se encontra.
application/xml	WHEN_NO_TEMPLATES	A solicitação é rejeitada com uma resposta HTTP 415 Unsupported Media Type.
application/xml	NEVER	A solicitação é rejeitada com uma resposta HTTP 415 Unsupported Media Type.

Exemplo 2: um modelo de mapeamento é definido na solicitação de integração para o tipo de conteúdo application/xml.

Content-type	Opção de passagem	Comportamento
Nenhum O padrão do API Gateway é application/json	WHEN_NO_MATCH	A carga da solicitação não é transformada e é enviada ao backend no estado em que se encontra.
Nenhum O padrão do API Gateway é application/json	WHEN_NO_TEMPLATES	A solicitação é rejeitada com uma resposta HTTP 415 Unsupported Media Type.
Nenhum O padrão do API Gateway é application/json	NEVER	A solicitação é rejeitada com uma resposta HTTP 415 Unsupported Media Type.
application/json	WHEN_NO_MATCH	A carga da solicitação não é transformada e é enviada ao backend no estado em que se encontra.
application/json	WHEN_NO_TEMPLATES	A solicitação é rejeitada com uma resposta HTTP 415 Unsupported Media Type.
application/json	NEVER	A solicitação é rejeitada com uma resposta HTTP 415 Unsupported Media Type.
application/xml	WHEN_NO_MATCH	A carga da solicitação é transformada usando o modelo.

Content-type	Opção de passagem	Comportamento
application/xml	WHEN_NO_TEMPLATES	A carga da solicitação é transformada usando o modelo.
application/xml	NEVER	A carga da solicitação é transformada usando o modelo.

Exemplo 3: nenhum modelo de mapeamento está definido na solicitação de integração.

Content-type	Opção de passagem	Comportamento
Nenhum O padrão do API Gateway é application/json	WHEN_NO_MATCH	A carga da solicitação não é transformada e é enviada ao backend no estado em que se encontra.
Nenhum O padrão do API Gateway é application/json	WHEN_NO_TEMPLATES	A carga da solicitação não é transformada e é enviada ao backend no estado em que se encontra.
Nenhum O padrão do API Gateway é application/json	NEVER	A solicitação é rejeitada com uma resposta HTTP 415 Unsupported Media Type.
application/json	WHEN_NO_MATCH	A carga da solicitação não é transformada e é enviada ao backend no estado em que se encontra.
application/json	WHEN_NO_TEMPLATES	A carga da solicitação não é transformada e é enviada ao

Content-type	Opção de passagem	Comportamento
		backend no estado em que se encontra.
application/json	NEVER	A solicitação é rejeitada com uma resposta HTTP 415 Unsupported Media Type.
application/xml	WHEN_NO_MATCH	A carga da solicitação não é transformada e é enviada ao backend no estado em que se encontra.
application/xml	WHEN_NO_TEMPLATES	A carga da solicitação não é transformada e é enviada ao backend no estado em que se encontra.
application/xml	NEVER	A solicitação é rejeitada com uma resposta HTTP 415 Unsupported Media Type.

Exemplo adicional de modelo de mapeamento para APIs REST no API Gateway

O exemplo a seguir mostra uma API de álbum de fotos no API Gateway que usa modelos de mapeamento para transformar dados de solicitação e de resposta de integração. Ele também usa modelos de dados para definir cargas úteis de solicitação de método e resposta de integração. Para saber mais sobre modelos de dados, consulte [Modelos de dados para APIs REST](#).

Solicitação de método e solicitação de integração

Veja a seguir um modelo que define o corpo da solicitação do método. Esse modelo de entrada exige que o chamador faça upload de uma página de foto requer no mínimo 10 fotos por página. É possível usar esse modelo de entrada para gerar um SDK ou para usar uma validação de solicitação para a API. Ao usar a validação de solicitação, se o corpo da solicitação do método não aderir à estrutura de dados do modelo, o API Gateway falhará na solicitação.

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

Veja a seguir um exemplo de corpo de solicitação de método que adere à estrutura de dados do modelo de dados 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
    }
  ]
}
```

Nesse exemplo, se o corpo da solicitação do método anterior foi enviado pelo cliente, esse modelo de mapeamento transforma a carga útil para corresponder ao formato exigido pelo endpoint de integração.

```
#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
]
```

Veja no exemplo a seguir os dados de saída da transformação.

```
{
  "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
    }
  ]
}
```

Esses dados são enviados para a solicitação de integração e, depois, ao endpoint de integração.

Resposta de integração e resposta do método

Veja a seguir um exemplo de modelo de saída para dados de fotos do endpoint de integração. Você pode usar esse modelo como um modelo de resposta de método, que é necessário quando você

gera um SDK fortemente tipado para a API. Isso faz com que a saída seja convertida em uma classe adequada em Java ou 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" }
        }
      }
    }
  }
}
```

O endpoint de integração pode não responder com uma resposta que faça adesão da estrutura de dados desse modelo. A resposta de integração pode parecer com o seguinte:

```
"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
  }
]
```

O exemplo de modelo de mapeamento a seguir transforma os dados de resposta de integração no formato esperado pela resposta do 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
  ]
}
```

Veja no exemplo a seguir os dados de saída da transformação.

```
{
  "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
  }
]
```

Esses dados são enviados para a resposta do método e, depois, de volta ao cliente.

Substituir parâmetros de solicitação e resposta da API e códigos de status por APIs REST no API Gateway

É possível usar transformações de modelos de mapeamento para substituir qualquer tipo de parâmetro de solicitação, cabeçalho de resposta ou código de status da resposta. Use um modelo de mapeamento para fazer o seguinte:

- Executar mapeamentos de parâmetros muitos para um.
- Substituir os parâmetros após a aplicação dos mapeamentos padrão do API Gateway
- Associar parâmetros condicionalmente com base no conteúdo do corpo ou outros valores de parâmetros.
- Criar parâmetros de modo programático.
- Substituir os códigos de status exibidos pelo seu endpoint de integração.

As substituições são feitas no final. Uma substituição só pode ser aplicada a um parâmetro por vez. Se você tentar substituir o mesmo parâmetro várias vezes, o API Gateway vai gerar uma resposta 5XX. Se você tiver de substituir o mesmo parâmetro várias vezes em todo o modelo, é recomendável criar uma variável e aplicar a substituição no final do modelo. O modelo é aplicado somente depois que todo o modelo é analisado.

Exemplo 1: substituir o código de status com base no corpo da integração

Veja a seguir como usar a [API de exemplo](#) para substituir o código de status com base no corpo da resposta de integração.

Console de gerenciamento da AWS

Como substituir um código de status com base no corpo da resposta de integração

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione Create API (Criar API).
3. Em API REST, escolha Criar.
4. Em Detalhes da API, escolha API de exemplo.
5. Selecione Criar API.

O API Gateway cria uma API de exemplo da loja de animais de estimação. Para recuperar informações sobre um animal de estimação, você usa a solicitação de método de API de GET `/pets/{petId}`, em que `{petId}` é um parâmetro de caminho correspondente a um número de ID de um animal de estimação.

Nesse exemplo, você substitui o código de resposta do método GET por 400 quando uma condição de erro é detectada.

6. Na árvore Recursos, selecione o método GET em `/petId`.
7. Primeiro, você testa a implementação atual da API.

Selecione a guia Testar. Talvez seja necessário selecionar o botão de seta para a direita para mostrar a guia.

8. Em `petId`, insira `-1` e, depois, selecione Testar.

O Corpo da resposta indica um erro fora do intervalo:

```
{
  "errors": [
    {
      "key": "GetPetRequest.petId",
      "message": "The value is out of range."
    }
  ]
}
```

Além disso, a última linha em Logs termina com: `Method completed with status: 200`.

A integração foi concluída com sucesso, mas houve um erro. Agora, você vai substituir o código de status com base na resposta de integração.

9. Na guia Resposta de integração, em Padrão - Resposta, selecione Editar.
10. Selecione Modelos de mapeamento.
11. Escolha Add mapping template (Adicionar modelo de mapeamento).
12. Em Tipo de conteúdo, insira **application/json**.
13. Em Corpo do modelo, insira o seguinte:

```
#set($inputRoot = $input.path('$'))
$input.json("$")
#if($inputRoot.toString().contains("error"))
#set($context.responseOverride.status = 400)
#end
```

Esse modelo de mapeamento usa a variável `$context.responseOverride.status` para substituir o código de status por 400 se a resposta de integração contiver a string `error`.

14. Escolha Salvar.
15. Selecione a guia Testar.
16. Em `petId`, insira **-1**.
17. Nos resultados, Response Body (Corpo da resposta) indica um erro fora do intervalo:

```
{
  "errors": [
    {
      "key": "GetPetRequest.petId",
      "message": "The value is out of range."
    }
  ]
}
```

Entretanto, a última linha em Logs agora termina com: `Method completed with status: 400`.

CloudFormation

Nesse exemplo, você usa a propriedade [corpo](#) para importar um arquivo de definição da OpenAPI para o API Gateway.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Body:
        openapi: 3.0.1
        info:
          title: PetStore Example 1
          description: Example pet store API.
          version: "2025-01-14T00:13:18Z"
        paths:
          /pets/{petId}:
            get:
              parameters:
                - name: petId
                  in: path
                  required: true
                  schema:
                    type: string
              responses:
                "200":
                  description: 200 response
              x-amazon-apigateway-integration:
                httpMethod: GET
                uri: http://petstore.execute-api.us-east-1.amazonaws.com/petstore/
            responses:
              default:
                statusCode: "200"
                responseTemplates:
                  application/json: |-
                    #set($inputRoot = $input.path('$'))
                    $input.json("$")
                    #if($inputRoot.toString().contains("error"))
                    #set($context.responseOverride.status = 400)
                    #end
              requestParameters:
                integration.request.path.petId: method.request.path.petId
```

```

        passthroughBehavior: when_no_match
        type: http
    components:
        schemas:
            Pet:
                type: object
                properties:
                    id:
                        type: integer
                    type:
                        type: string
                    price:
                        type: number
    ApiGatewayDeployment:
        Type: 'AWS::ApiGateway::Deployment'
        DependsOn: Api
        Properties:
            RestApiId: !Ref Api
    ApiGatewayDeployment20250219:
        Type: 'AWS::ApiGateway::Deployment'
        DependsOn: Api
        Properties:
            RestApiId: !Ref Api
    Stage:
        Type: 'AWS::ApiGateway::Stage'
        Properties:
            DeploymentId: !Ref ApiGatewayDeployment20250219
            RestApiId: !Ref Api
            StageName: prod

```

OpenAPI

A definição de OpenAPI a seguir cria o recurso GET `pets/{petId}` e substitui o código de status com base no corpo da integração.

```

{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "PetStore Example 1",
    "description" : "Example pet store API.",
    "version" : "2025-01-14T00:13:18Z"
  },
  "paths" : {

```



```

"/pets/{petId}" : {
  "get" : {
    "parameters" : [ {
      "name" : "petId",
      "in" : "path",
      "required" : true,
      "schema" : {
        "type" : "string"
      }
    } ],
    "responses" : {
      "200" : {
        "description" : "200 response"
      }
    },
    "x-amazon-apigateway-integration" : {
      "httpMethod" : "GET",
      "uri" : "http://petstore.execute-api.us-east-1.amazonaws.com/petstore/pets/{petId}",
      "responses" : {
        "default" : {
          "statusCode" : "200",
          "responseTemplates" : {
            "application/json" : "#set($inputRoot = $input.path('$'))\n\n$input.json(\"$\")\n\n#if($inputRoot.toString().contains(\"error\n\"))\n\n#set($context.responseOverride.status = 400)\n\n#end"
          }
        }
      },
      "requestParameters" : {
        "integration.request.path.petId" : "method.request.path.petId"
      },
      "passthroughBehavior" : "when_no_match",
      "type" : "http"
    }
  }
},
"components" : {
  "schemas" : {
    "Pet" : {
      "type" : "object",
      "properties" : {
        "id" : {

```

```
        "type" : "integer"
      },
      "type" : {
        "type" : "string"
      },
      "price" : {
        "type" : "number"
      }
    }
  }
}
```

Exemplo 2: substituir o cabeçalho da solicitação e criar outros cabeçalhos

O exemplo a seguir usa a [API de exemplo](#) para substituir o cabeçalho da solicitação e criar outros cabeçalhos.

Console de gerenciamento da AWS

Como substituir o cabeçalho de solicitação de um método criando outro cabeçalho

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha a API de exemplo que você criou no tutorial anterior. O nome da API deve ser PetStore.
3. Na árvore Recursos, selecione o método GET em /pet.
4. Na guia Solicitação de método, em Configurações de solicitação de método, selecione Editar.
5. Selecione Cabeçalhos de solicitação HTTP e, depois, Adicionar cabeçalho.
6. Em Nome, digite **header1**.
7. Selecione Adicionar cabeçalho e, depois, crie um segundo cabeçalho chamado **header2**.
8. Escolha Salvar.

Agora, você combina esses cabeçalhos em um valor de cabeçalho usando um modelo de mapeamento.

9. Na guia Solicitação de integração, em Configurações de solicitação de integração, selecione Editar.

10. Em Passagem do corpo da solicitação, selecione Quando não há modelos definidos (recomendado).
11. Selecione Modelos de mapeamento e, depois, faça o seguinte:
 - a. Escolha Add mapping template (Adicionar modelo de mapeamento).
 - b. Em Tipo de conteúdo, insira **application/json**.
 - c. Em Corpo do modelo, insira o seguinte:

```
#set($header10override = "pets")
#set($header3Value = "$input.params('header1')$input.params('header2')")
$input.json("$")
#set($context.request0override.header.header3 = $header3Value)
#set($context.request0override.header.header1 = $header10override)
#set($context.request0override.header.multivalueheader=[$header10override,
$header3Value])
```

Esse modelo de mapeamento substitui header1 pela string pets e cria um cabeçalho de vários valores chamado \$header3Value que combina header1 e header2.

12. Escolha Salvar.
13. Selecione a guia Testar.
14. Em Cabeçalhos, copie o seguinte código:

```
header1:header1Val
header2:header2Val
```

15. Escolha Test (Testar).

Em Logs, você deve ver uma entrada que inclui 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

Nesse exemplo, você usa a propriedade [corpo](#) para importar um arquivo de definição da OpenAPI para o API Gateway.

```

AWSTemplateFormatVersion: 2010-09-09
Resources:
  Api:
    Type: 'AWS::ApiGateway::RestApi'
    Properties:
      Body:
        openapi: 3.0.1
        info:
          title: PetStore Example 2
          description: Example pet store API.
          version: "2025-01-14T00:36:18Z"
        paths:
          /pets:
            get:
              parameters:
                - name: header2
                  in: header
                  schema:
                    type: string
                - name: page
                  in: query
                  schema:
                    type: string
                - name: type
                  in: query
                  schema:
                    type: string
                - name: header1
                  in: header
                  schema:
                    type: string
              responses:
                "200":
                  description: 200 response
              x-amazon-apigateway-integration:
                httpMethod: GET
                uri: http://petstore.execute-api.us-east-1.amazonaws.com/petstore/
pets
              responses:
                default:
                  statusCode: "200"
              requestParameters:
                integration.request.header.header1: method.request.header.header1

```

```

        integration.request.header.header2: method.request.header.header2
        integration.request.querystring.page:
method.request.querystring.page
        integration.request.querystring.type:
method.request.querystring.type
    requestTemplates:
      application/json: |-
        #set($header10override = "pets")
        #set($header3Value =
"$input.params('header1')$input.params('header2')")
        $input.json("$")
        #set($context.requestOverride.header.header3 = $header3Value)
        #set($context.requestOverride.header.header1 = $header10override)

#set($context.requestOverride.header.multivalueheader=[$header10override,
$header3Value])
    passthroughBehavior: when_no_match
    type: http
  components:
    schemas:
      Pet:
        type: object
        properties:
          id:
            type: integer
          type:
            type: string
          price:
            type: number
  ApiGatewayDeployment:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn: Api
    Properties:
      RestApiId: !Ref Api
  ApiGatewayDeployment20250219:
    Type: 'AWS::ApiGateway::Deployment'
    DependsOn: Api
    Properties:
      RestApiId: !Ref Api
  Stage:
    Type: 'AWS::ApiGateway::Stage'
    Properties:
      DeploymentId: !Ref ApiGatewayDeployment20250219
      RestApiId: !Ref Api

```

```
StageName: prod
```

OpenAPI

A definição de OpenAPI a seguir cria o recurso GET `pets`, substitui o cabeçalho da solicitação e cria outros cabeçalhos.

```
{
  "openapi" : "3.0.1",
  "info" : {
    "title" : "PetStore Example 2",
    "description" : "Example pet store API.",
    "version" : "2025-01-14T00:36:18Z"
  },
  "paths" : {
    "/pets" : {
      "get" : {
        "parameters" : [ {
          "name" : "header2",
          "in" : "header",
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "page",
          "in" : "query",
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "type",
          "in" : "query",
          "schema" : {
            "type" : "string"
          }
        }, {
          "name" : "header1",
          "in" : "header",
          "schema" : {
            "type" : "string"
          }
        }
      ],
      "responses" : {
```

```

        "200" : {
            "description" : "200 response"
        }
    },
    "x-amazon-apigateway-integration" : {
        "httpMethod" : "GET",
        "uri" : "http://petstore.execute-api.us-east-1.amazonaws.com/petstore/
pets",
        "responses" : {
            "default" : {
                "statusCode" : "200"
            }
        },
        "requestParameters" : {
            "integration.request.header.header1" : "method.request.header.header1",
            "integration.request.header.header2" : "method.request.header.header2",
            "integration.request.querystring.page" :
"method.request.querystring.page",
            "integration.request.querystring.type" :
"method.request.querystring.type"
        },
        "requestTemplates" : {
            "application/json" : "#set($header10override = \"pets
\\\")\n#set($header3Value = \"$input.params('header1')$input.params('header2')\\\")\n
$input.json(\"$\")\n#set($context.request0override.header.header3
= $header3Value)\n#set($context.request0override.header.header1 =
$header10override)\n#set($context.request0override.header.multivalueheader=[$header10override,
$header3Value])"
        },
        "passthroughBehavior" : "when_no_match",
        "type" : "http"
    }
}
}
}
}
}

```

Para usar uma substituição de modelo de mapeamento, adicione uma ou mais das variáveis \$context a seguir. Consulte uma lista de variáveis \$context em [the section called “Variáveis de contexto para transformações de dados”](#).

Tutorial: modificar a solicitação e a resposta de integração para integrações a serviços da AWS

O tutorial a seguir mostra como usar transformações de modelos de mapeamento para configurar modelos de mapeamento a fim de transformar solicitações e respostas de integração usando o console e a AWS CLI.

Tópicos

- [Configurar transformações de dados no console do API Gateway](#)
- [Configurar a transformação de dados usando a CLI da AWS](#)
- [Modelo CloudFormation de transformação de dados concluído](#)

Configurar transformações de dados no console do API Gateway

Neste tutorial, você criará uma API incompleta e uma tabela do DynamoDB usando o seguinte arquivo .zip [data-transformation-tutorial-console.zip](#). Essa API incompleta tem um recurso / pets com os métodos GET e POST.

- O método GET obterá dados do endpoint HTTP `http://petstore-demo-endpoint.execute-api.com/petstore/pets`. Os dados de saída serão transformados de acordo com o modelo de mapeamento em [the section called “Transformações de modelo de mapeamento”](#).
- O método POST possibilitará que o usuário POST informações sobre animais de estimação em uma tabela do Amazon DynamoDB usando um modelo de mapeamento.

Baixe e descompacte [o modelo de criação de aplicativos para CloudFormation](#). Você usará esse modelo para criar uma tabela do DynamoDB para publicar informações sobre animais de estimação e uma API incompleta. Você concluirá o restante das etapas no console do API Gateway.

Como criar uma pilha do CloudFormation

1. Abra o console do CloudFormation em <https://console.aws.amazon.com/cloudformation>.
2. Selecione Create stack (Criar pilha) e With new resources (standard) (Com novos recursos (padrão)).
3. Em Specify template (Especificar modelo), escolha Upload a template file (Fazer upload de um arquivo de modelo).
4. Selecione o modelo que você baixou.

5. Escolha Next (Próximo).
6. Em Nome da pilha, insira **data-transformation-tutorial-console** e escolha Avançar.
7. Para Configurar opções de pilha, escolha Avançar.
8. Para Capabilities (Recursos), reconheça que CloudFormation pode criar recursos do IAM em sua conta.
9. Escolha Próximo e, depois, Enviar.

CloudFormationO fornece os recursos especificados no modelo. Pode demorar alguns minutos para concluir o provisionamento de seus recursos. Quando o status da sua pilha do CloudFormation for CREATE_COMPLETE, você estará pronto para passar para a próxima etapa.

Como testar a resposta de integração **GET**

1. Na guia Recursos da pilha de CloudFormation para **data-transformation-tutorial-console**, selecione o ID físico de sua API.
2. No painel de navegação, selecione Recursos e, depois, selecione o método GET.
3. Selecione a guia Testar. Talvez seja necessário selecionar o botão de seta para a direita para mostrar a guia.

A saída do teste mostrará o seguinte:

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

Você transformará esse resultado de acordo com o modelo de mapeamento em [the section called “Transformações de modelo de mapeamento”](#).

Como transformar a resposta de integração **GET**

1. Selecione a guia Resposta de integração.

Atualmente, não há modelos de mapeamento definidos, portanto, a resposta de integração não será transformada.

2. Em Padrão - Resposta, selecione Editar.
3. Selecione Modelos de mapeamento e, depois, faça o seguinte:
 - a. Escolha Add mapping template (Adicionar modelo de mapeamento).
 - b. Em Tipo de conteúdo, insira **application/json**.
 - c. Em Corpo do modelo, insira o seguinte:

```
#set($inputRoot = $input.path('$'))
[
  #foreach($elem in $inputRoot)
    {
      "description" : "Item $elem.id is a $elem.type.",
      "askingPrice" : $elem.price
    }#if($foreach.hasNext),#end
  #end
]
```

Escolha Salvar.

Como testar a resposta de integração **GET**

- Selecione a guia Testar e, depois, Testar.

A saída do teste mostrará a resposta 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
  }
]
```

Como transformar dados de entrada do método **POST**

1. Selecione o método POST.
2. Selecione a guia Solicitação de integração e, em Configurações de solicitação de integração, selecione Editar.

O modelo CloudFormation preencheu alguns dos campos da solicitação de integração.

- O tipo de integração é AWS service (Serviço da AWS).
- O AWS service (Serviço da AWS) é o DynamoDB.
- O método HTTP é o POST.
- A ação é PutItem.
- O perfil de execução que possibilita que o API Gateway coloque um item na tabela do DynamoDB é `data-transformation-tutorial-console-APIGatewayRole`.
O CloudFormation criou esse perfil para possibilitar que o API Gateway tenha as permissões mínimas para interagir com o DynamoDB.

O nome da tabela do DynamoDB não foi especificado. Você especificará o nome nas etapas a seguir.

3. Em Passagem do corpo da solicitação, selecione Nunca.

Isso significa que a API rejeitará dados com tipos de conteúdo que não tenham um modelo de mapeamento.

4. Selecione Modelos de mapeamento.

5. O Tipo de conteúdo é definido como `application/json`. Isso significa que tipos de conteúdo que não sejam `application/json` serão rejeitados pela API. Para obter mais informações sobre os comportamentos de passagem direta de integração, consulte [Comportamento de solicitação de método para cargas úteis sem modelos de mapeamento para APIs REST no API Gateway](#)
6. Insira o código a seguir no 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")
    }
  }
}
```

Esse modelo especifica a tabela como `data-transformation-tutorial-console-ddb` e define os itens como `id`, `type` e `price`. Os itens virão do corpo do método **POST**. Você também pode usar um modelo de dados para ajudar a criar um modelo de mapeamento. Para obter mais informações, consulte [Solicitar validação para APIs REST no API Gateway](#).

7. Escolha Salvar para salvar seu modelo de mapeamento.

Como adicionar um método e uma resposta de integração do método **POST**

O CloudFormation criou um método em branco e uma resposta de integração. Você editará essa resposta para fornecer mais informações. Para ter mais informações sobre como editar respostas, consulte [Exemplos de mapeamento de parâmetros para APIs REST no API Gateway](#).

1. Na guia Resposta de integração, em Padrão - Resposta, selecione Editar.
2. Selecione Modelos de mapeamento e, depois, Adicionar modelo de mapeamento.
3. Em Tipo de conteúdo, insira **`application/json`**.
4. No editor de código, insira o seguinte modelo de mapeamento de saída para enviar uma mensagem de saída:

```
{ "message" : "Your response was recorded at $context.requestTime" }
```

Para ter mais informações sobre variáveis de contexto, consulte [Variáveis de contexto para transformações de dados](#).

5. Escolha Salvar para salvar seu modelo de mapeamento.

Testar o método **POST**

Selecione a guia Testar. Talvez seja necessário selecionar o botão de seta para a direita para mostrar a guia.

1. No corpo da solicitação, insira o exemplo a seguir.

```
{
    "id": "4",
    "type" : "dog",
    "price": "321"
}
```

2. Escolha Testar.

A saída deve mostrar sua mensagem de êxito.

Você pode abrir o console do DynamoDB em <https://console.aws.amazon.com/dynamodb/> para verificar se o item de exemplo está na sua tabela.

Para excluir uma pilha do CloudFormation

1. Abra o console do CloudFormation em <https://console.aws.amazon.com/cloudformation>.
2. Selecione sua pilha do CloudFormation.
3. Escolha Excluir e, em seguida, confirme sua escolha.

Configurar a transformação de dados usando a CLI da AWS

Neste tutorial, você criará uma API incompleta e uma tabela do DynamoDB usando o arquivo [.zip data-transformation-tutorial-console.zip](#) a seguir. Essa API incompleta tem um recurso `/pets` com um método GET integrado ao endpoint HTTP `http://petstore-demo-`

`endpoint.execute-api.com/petstore/pets`. Você criará um método POST para se conectar a uma tabela do DynamoDB e usará modelos de mapeamento para inserir dados em uma tabela do DynamoDB.

- Você transformará os dados de saída de acordo com o modelo de mapeamento em [the section called “Transformações de modelo de mapeamento”](#).
- Você criará um método POST para possibilitar que o usuário POST informações sobre animais de estimação em uma tabela do Amazon DynamoDB usando um modelo de mapeamento.

Como criar uma pilha do CloudFormation

Baixe e descompacte [o modelo de criação de aplicativos para CloudFormation](#).

Para concluir o tutorial a seguir, é necessária a [AWS Command Line Interface \(AWS CLI\) versão 2](#).

Para comandos longos, um caractere de escape (\) é usado para dividir um comando em várias linhas.

Note

No Windows, alguns comandos da CLI do Bash que você costuma usar (como `zip`) não são compatíveis com os terminais integrados do sistema operacional. Para obter uma versão do Ubuntu com o Bash integrada no Windows, [instale o Subsistema do Windows para Linux](#). Os exemplos de comandos da CLI neste guia usam a formatação Linux. Os comandos que incluem documentos JSON em linha deverão ser reformatados se você estiver usando a CLI do Windows.

1. Use o comando a seguir para criar a pilha de CloudFormation.

```
aws cloudformation create-stack --stack-name data-transformation-tutorial-cli
--template-body file://data-transformation-tutorial-cli.zip --capabilities
CAPABILITY_NAMED_IAM
```

2. CloudFormationO fornece os recursos especificados no modelo. Pode demorar alguns minutos para concluir o provisionamento de seus recursos. Use o comando a seguir para ver o status de sua pilha de CloudFormation.

```
aws cloudformation describe-stacks --stack-name data-transformation-tutorial-cli
```

3. Quando o status da sua pilha de CloudFormation for `StackStatus: "CREATE_COMPLETE"`, use o comando a seguir para recuperar valores de saída relevantes para etapas futuras.

```
aws cloudformation describe-stacks --stack-name data-transformation-tutorial-cli  
--query "Stacks[*].Outputs[*].{OutputKey: OutputKey, OutputValue: OutputValue,  
Description: Description}"
```

Os valores de saída são os seguintes:

- `ApiRole`, que é o nome do perfil que possibilita que o API Gateway coloque itens na tabela do DynamoDB. Para este tutorial, o nome do perfil é `data-transformation-tutorial-cli-APIGatewayRole-ABCDEFGH`.
- `DDBTableName`, que é o nome da tabela do DynamoDB. Para este tutorial, o nome da tabela é `data-transformation-tutorial-cli-ddb`.
- `ResourceId`, que é o ID do recurso de animais de estimação em que os métodos GET e POST são expostos. Para este tutorial, o ID do recurso é `efg456`.
- `ApiId`, que é o ID da API. Para este tutorial, o ID da API é `abc123`.

Como testar o método **GET** antes da transformação de dados

- Use o comando a seguir para testar o método GET.

```
aws apigateway test-invoke-method --rest-api-id abc123 \  
--resource-id efg456 \  
--http-method GET
```

A saída do teste mostrará o seguinte:

```
[  
  {  
    "id": 1,  
    "type": "dog",  
    "price": 249.99  
  },  
  {  
    "id": 2,  
    "type": "cat",  
    "price": 124.99  
  },  
]
```

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

Você transformará esse resultado de acordo com o modelo de mapeamento em [the section called “Transformações de modelo de mapeamento”](#).

Como transformar a resposta de integração **GET**

- Use o comando a seguir para atualizar a resposta de integração do método GET. Substitua o *rest-api-id* e o *resource-id* pelos seus valores.

Use o comando a seguir para criar uma resposta de integração.

```
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\\n\"}"}'
```

Como testar o método **GET**

- Use o comando a seguir para testar o método GET.

```
aws apigateway test-invoke-method --rest-api-id abc123 \
  --resource-id efg456 \
  --http-method GET \
```

A saída do teste mostrará a resposta 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
  }
]
```

Como criar um método **POST**

1. Use o comando a seguir para criar um novo método no recurso `/pets`.

```
aws apigateway put-method --rest-api-id abc123 \  
  --resource-id efg456 \  
  --http-method POST \  
  --authorization-type "NONE" \
```

Esse método possibilitará que você envie informações sobre animais de estimação para a tabela do DynamoDB que você criou na pilha de CloudFormation.

2. Use o comando a seguir para criar uma integração do AWS service (Serviço da AWS) no 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-ABCDEFGF \  
  --request-templates '{"application/json":{"\n\"TableName\":\n\"data-transformation-tutorial-cli-ddb\", \"Item\":{\n\"id\":{\n\"N\":$input.json(\n\"$.id\")\"},\n\"type\":{\n\"S\":$input.json(\n\"$.type\")\"},\n\"price\":{\n\"N\":$input.json(\n\"$.price\")\"} } }"}'
```

3. Use o comando a seguir para criar uma resposta de método para uma chamada bem-sucedida do método POST.

```
aws apigateway put-method-response --rest-api-id abc123 \  
  --resource-id efg456 \  
  --http-method POST \  
  --status-code 200
```

4. Use o comando a seguir para criar uma resposta de integração para uma chamada bem-sucedida do 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\\"}"}'
```

Como testar o método **POST**

- Use o comando a seguir para testar o método POST.

```
aws apigateway test-invoke-method --rest-api-id abc123 \  
  --resource-id efg456 \  
  --http-method POST \  
  --body '{"id": "4", "type": "dog", "price": "321"}'
```

A saída mostrará a mensagem de êxito.

Para excluir uma pilha do CloudFormation

- Use o comando a seguir para excluir seus recursos de CloudFormation.

```
aws cloudformation delete-stack --stack-name data-transformation-tutorial-cli
```

Modelo CloudFormation de transformação de dados concluído

O exemplo a seguir é um modelo de CloudFormation completo, que cria uma API e uma tabela do DynamoDB com um recurso `/pets` com os métodos GET e POST.

- O método GET obterá dados do endpoint HTTP `http://petstore-demo-endpoint.execute-api.com/petstore/pets`. Os dados de saída serão transformados de acordo com o modelo de mapeamento em [the section called “Transformações de modelo de mapeamento”](#).
- O método POST possibilitará que o usuário POST informações sobre animais de estimação em uma tabela do DynamoDB usando um modelo de mapeamento.

Exemplo de modelo CloudFormation

```
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
```

Exemplos de uso de variáveis para mapear transformações de modelos para o API Gateway

Os exemplos a seguir mostram como usar as variáveis `$context`, `input` e `util` em modelos de mapeamento. Você pode usar uma integração simulada ou uma integração sem proxy do Lambda que retorna o evento de entrada ao API Gateway. Consulte uma lista de variáveis compatíveis com transformadores de dados em [the section called “Variáveis para transformações de dados”](#).

Exemplo 1: passar várias variáveis **\$context** ao endpoint de integração

O exemplo a seguir mostra um modelo de mapeamento que mapeia variáveis `$context` de entrada para variáveis de backend com nomes um pouco diferentes em uma carga de solicitação de integração:

```
{
  "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"
}
```

A saída deste modelo de mapeamento deve ser semelhante ao seguinte:

```
{
  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'
}
```

Uma das variáveis é uma chave de API. Este exemplo pressupõe que o método exige a chave de API.

Exemplo 2: passar todos os parâmetros de solicitação ao endpoint de integração por meio de uma carga útil JSON:

O exemplo a seguir passa todos os parâmetros de solicitação, incluindo os parâmetros path, querystring e header, ao endpoint de integração por meio de uma carga útil 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
  }
}
```

Se uma solicitação tiver estes parâmetros de entrada:

- Um parâmetro de caminho chamado myparam
- Parâmetros de string de consulta querystring1=value1,value2
- Cabeçalhos "header1" : "value1".

A saída deste modelo de mapeamento deve ser semelhante ao seguinte:

```
{"params":{"path":{"example2":"myparamm"},"querystring":  
{"querystring1":"value1,value2"},"header":{"header1":"value1"}}}
```

Exemplo 3: passar uma subseção de uma solicitação de método para o endpoint de integração

O exemplo a seguir usa o parâmetro de entrada `name` para recuperar somente o parâmetro `name` e o parâmetro de entrada `input.json('$')` para recuperar todo o corpo da solicitação do método:

```
{  
  "name" : "$input.params('name')",  
  "body" : $input.json('$')  
}
```

Para uma solicitação que inclua os parâmetros de string de consulta `name=Bella&type=dog` e o seguinte corpo:

```
{  
  "Price" : "249.99",  
  "Age": "6"  
}
```

A saída deste modelo de mapeamento deve ser semelhante ao seguinte:

```
{  
  "name" : "Bella",  
  "body" : {"Price":"249.99","Age":"6"}  
}
```

Esse modelo de mapeamento remove o parâmetro da string de consulta `type=dog`.

Se a entrada JSON contiver caracteres sem escape que não podem ser analisados pelo JavaScript, o API Gateway pode retornar uma resposta 400. Aplique `$util.escapeJavaScript($input.json('$'))` para garantir que a entrada JSON possa ser analisada corretamente.

O exemplo anterior com `$util.escapeJavaScript($input.json('$'))` aplicado é o seguinte:

```
{  
  "name" : "$input.params('name')",  
  "body" : "$util.escapeJavaScript($input.json('$'))"
```



```
}
```

Neste caso, a saída deste modelo de mapeamento deve ser semelhante ao seguinte:

```
{
  "name" : "Bella",
  "body": {"Price":"249.99","Age":"6"}
}
```

Exemplo 4: usar a expressão JSONPath para passar uma subseção de uma solicitação de método ao endpoint de integração

O exemplo a seguir usa expressões JSONPath para recuperar somente o parâmetro de entrada name e o Age do corpo da solicitação:

```
{
  "name" : "$input.params('name')",
  "body" : $input.json('$.Age')
}
```

Para uma solicitação que inclua os parâmetros de string de consulta name=Bella&type=dog e o seguinte corpo:

```
{
  "Price" : "249.99",
  "Age": "6"
}
```

A saída deste modelo de mapeamento deve ser semelhante ao seguinte:

```
{
  "name" : "Bella",
  "body" : "6"
}
```

Esse modelo de mapeamento remove o parâmetro da string de consulta type=dog e o campo Price do corpo.

Se uma carga útil de solicitação de método contiver caracteres sem escape que não podem ser analisados por JavaScript, o API Gateway pode retornar uma resposta 400. Aplique

`$util.escapeJavaScript()` para garantir que a entrada JSON possa ser analisada corretamente.

O exemplo anterior com `$util.escapeJavaScript($input.json('$.Age'))` aplicado é o seguinte:

```
{
  "name" : "$input.params('name')",
  "body" : "$util.escapeJavaScript($input.json('$.Age'))"
}
```

Neste caso, a saída deste modelo de mapeamento deve ser semelhante ao seguinte:

```
{
  "name" : "Bella",
  "body": "\"6\""
}
```

Exemplo 5: usar uma expressão JSONPath para passar informações sobre uma solicitação de método ao endpoint de integração

O exemplo a seguir usa `$input.params()`, `$input.path()` e `$input.json()` para enviar informações sobre uma solicitação de método ao endpoint de integração. Esse modelo de mapeamento usa o método `size()` para fornecer o número de elementos em uma lista.

```
{
  "id" : "$input.params('id')",
  "count" : "$input.path('$.things').size()",
  "things" : $input.json('$.things')
}
```

Para uma solicitação que inclui o parâmetro de caminho 123 e o seguinte corpo:

```
{
  "things": {
    "1": {},
    "2": {},
    "3": {}
  }
}
```

A saída deste modelo de mapeamento deve ser semelhante ao seguinte:

```
{"id":"123","count":"3","things":{"1":{},"2":{},"3":{}}}
```

Se uma carga útil de solicitação de método contiver caracteres sem escape que não podem ser analisados por JavaScript, o API Gateway pode retornar uma resposta 400. Aplique `$util.escapeJavaScript()` para garantir que a entrada JSON possa ser analisada corretamente.

O exemplo anterior com `$util.escapeJavaScript($input.json('$.things'))` aplicado é o seguinte:

```
{
  "id" : "$input.params('id')",
  "count" : "$input.path('$.things').size()",
  "things" : "$util.escapeJavaScript($input.json('$.things'))"
}
```

A saída deste modelo de mapeamento deve ser semelhante ao seguinte:

```
{"id":"123","count":"3","things":{"1\":"{}","2\":"{}","3\":"{}}"}
```

Variáveis para transformações de dados para o API Gateway

Ao criar um mapeamento de parâmetros, é possível usar variáveis de contexto como a fonte de dados. Ao criar transformações de modelos de mapeamento, é possível usar variáveis de utilidade, entrada e contexto em scripts que você escreve na [Velocity Template Language \(VTL\)](#). Consulte exemplos de modelos de mapeamento que usam essas variáveis de referência em [the section called “Exemplos de uso de variáveis para mapear transformações de modelos”](#).

Consulte uma lista de variáveis de referência para registro em log de acesso em [the section called “Variáveis para registro em log de acesso no API Gateway”](#).

Variáveis de contexto para transformações de dados

É possível usar as variáveis `$context` a seguir para transformações de dados.

Parâmetro	Descrição
<code>\$context.accountId</code>	O ID da conta da AWS do proprietário da API

Parâmetro	Descrição
<code>\$context.apiId</code>	O identificador que o API Gateway atribui à sua API.
<code>\$context.authorizer.claims. <i>property</i></code>	Uma propriedade das declarações retornada s do grupo de usuários do Amazon Cognito depois que o autor da chamada do método é autenticado com êxito. Para obter mais informações, consulte the section called “Use o grupo de usuários do Amazon Cognito como um autorizador para uma API REST”.  Note Chamar <code>\$context.authorizer.claims</code> retorna um valor nulo.
<code>\$context.authorizer.principalId</code>	A identificação do usuário principal associada ao token enviado pelo cliente e retornada a partir de um autorizador do Lambda do API Gateway (anteriormente conhecido como autorizador personalizado). Para obter mais informações, consulte the section called “Usar os autorizadores do Lambda” .

Parâmetro	Descrição
<code>\$context.authorizer.</code> <i>property</i>	O valor transformado em string do par de chave/valor especificado do mapa context retornado de uma função de autorizador do Lambda do API Gateway. Por exemplo, se o autorizador retornar o seguinte mapa context: <pre>"context" : { "key": "value", "numKey": 1, "boolKey": true }</pre> Chamar <code>\$context.authorizer.key</code> retornará a string "value", chamar <code>\$context.authorizer.numKey</code> retornará a string "1" e chamar <code>\$context.authorizer.boolKey</code> retornará a string "true". Com relação à <i>propriedade</i> , o único caractere especial aceito é o sublinhado (<code>_</code>). Para obter mais informações, consulte the section called “Usar os autorizadores do Lambda”.
<code>\$context.awsEndpointRequestId</code>	O ID da solicitação do endpoint da AWS.
<code>\$context.deploymentId</code>	O ID da implantação de API.
<code>\$context.domainName</code>	O nome de domínio completo usado para invocar a API. Ele deve ser o mesmo que o cabeçalho Host de entrada.
<code>\$context.domainPrefix</code>	O primeiro rótulo do <code>\$context.domainName</code> e <code>.</code>

Parâmetro	Descrição
<code>\$context.error.message</code>	Uma string que contém uma mensagem de erro do API Gateway. Essa variável pode ser usada apenas para substituição de variável simples em um modelo de mapeamento de corpo GatewayResponse , que não é processado pelo mecanismo Velocity Template Language, bem como no registro de acesso. Para obter mais informações, consulte the section called “Métricas” e the section called “Configurar respostas do gateway para personalizar respostas de erro” .
<code>\$context.error.messageString</code>	O valor citado de <code>\$context.error.message</code> , ou seja, " <code>\$context.error.message</code> ".
<code>\$context.error.responseType</code>	Um tipo de GatewayResponse . Essa variável pode ser usada apenas para substituição de variável simples em um modelo de mapeamento de corpo GatewayResponse , que não é processado pelo mecanismo Velocity Template Language, bem como no registro de acesso. Para obter mais informações, consulte the section called “Métricas” e the section called “Configurar respostas do gateway para personalizar respostas de erro” .
<code>\$context.error.validationErrorString</code>	Uma string que contém uma mensagem de erro de validação detalhada.
<code>\$context.extendedRequestId</code>	O ID estendido que o API Gateway gera e atribui à solicitação de API. O ID de solicitação estendido contém informações úteis para a depuração e solução de problemas.

Parâmetro	Descrição
<code>\$context.httpMethod</code>	O método HTTP utilizado. Os valores válidos incluem: DELETE, GET, HEAD, OPTIONS, PATCH, POST e PUT.
<code>\$context.identity.accountId</code>	O ID da conta da AWS associado à solicitação.
<code>\$context.identity.apiKey</code>	Para os métodos de API que exigem uma chave de API, essa variável é a chave de API associada à solicitação do método. Para métodos que não exigem uma chave de API, essa variável é um valor nulo. Para obter mais informações, consulte the section called “Planos de uso” .
<code>\$context.identity.apiKeyId</code>	O ID da chave de API associada a uma solicitação de API que exige uma chave de API.
<code>\$context.identity.caller</code>	O identificador principal da entidade do autor da chamada que assinou a solicitação. Compatível com recursos que usam a autorização do IAM.

Parâmetro	Descrição
<code>\$context.identity.cognitoAuthenticationProvider</code>	Uma lista separada por vírgulas de todos os provedores de autenticação do Amazon Cognito usados pelo autor da chamada que faz a solicitação. Disponível somente se a solicitação foi assinada com credenciais do Amazon Cognito. Por exemplo, para uma identidade e de um grupo de usuários do Amazon Cognito, <code>cognito-idp.<i>region</i>.amazonaws.com/<i>user_pool_id</i></code>, <code>cognito-idp.<i>region</i>.amazonaws.com/<i>user_pool_id</i>:CognitoSignIn:<i>token subject claim</i></code> Consulte informações sobre os provedores de autenticação do Amazon Cognito disponíveis em Using Federated Identities no Guia do desenvolvedor do Amazon Cognito.
<code>\$context.identity.cognitoAuthenticationType</code>	O tipo de autenticação do Amazon Cognito do autor da chamada que faz a solicitação. Disponível somente se a solicitação foi assinada com credenciais do Amazon Cognito. Os valores possíveis incluem <code>authenticated</code> para identidades autenticadas e <code>unauthenticated</code> para identidades não autenticadas.
<code>\$context.identity.cognitoIdentityId</code>	O ID de identidade do Amazon Cognito do autor da chamada que faz a solicitação. Disponível somente se a solicitação foi assinada com credenciais do Amazon Cognito.

Parâmetro	Descrição
<code>\$context.identity.cognitoId</code> <code>entityPoolId</code>	O ID do grupo de identidades do Amazon Cognito do autor da chamada que faz a solicitação. Disponível somente se a solicitação foi assinada com credenciais do Amazon Cognito.
<code>\$context.identity.principalOrgId</code>	O ID da organização da AWS .
<code>\$context.identity.sourceIp</code>	O endereço IP de origem da conexão TCP mais próxima que está fazendo a solicitação ao endpoint do API Gateway.
<code>\$context.identity.clientCert</code> <code>t.clientCertPem</code>	O certificado de cliente codificado por PEM que o cliente apresentou durante a autenticação TLS mútua. Presente quando um cliente acessa uma API usando um nome de domínio personalizado que tenha TLS mútuo habilitado. Presente somente nos logs de acesso, se ocorrer falha na autenticação TLS mútua.
<code>\$context.identity.clientCert</code> <code>t.subjectDN</code>	O nome distinto do assunto do certificado apresentado por um cliente. Presente quando um cliente acessa uma API usando um nome de domínio personalizado que tenha TLS mútuo habilitado. Presente somente nos logs de acesso, se ocorrer falha na autenticação TLS mútua.
<code>\$context.identity.clientCert</code> <code>t.issuerDN</code>	O nome distinto do emissor do certificado apresentado por um cliente. Presente quando um cliente acessa uma API usando um nome de domínio personalizado que tenha TLS mútuo habilitado. Presente somente nos logs de acesso, se ocorrer falha na autenticação TLS mútua.

Parâmetro	Descrição
<code>\$context.identity.clientCertificate.serialNumber</code>	O número de série do certificado. Presente quando um cliente acessa uma API usando um nome de domínio personalizado que tenha TLS mútuo habilitado. Presente somente nos logs de acesso, se ocorrer falha na autenticação TLS mútua.
<code>\$context.identity.clientCertificate.validity.notBefore</code>	A data antes da qual o certificado é inválido. Presente quando um cliente acessa uma API usando um nome de domínio personalizado que tenha TLS mútuo habilitado. Presente somente nos logs de acesso, se ocorrer falha na autenticação TLS mútua.
<code>\$context.identity.clientCertificate.validity.notAfter</code>	A data após a qual o certificado é inválido. Presente quando um cliente acessa uma API usando um nome de domínio personalizado que tenha TLS mútuo habilitado. Presente somente nos logs de acesso, se ocorrer falha na autenticação TLS mútua.
<code>\$context.identity.vpcId</code>	O ID da VPC que está fazendo a solicitação ao endpoint do API Gateway.
<code>\$context.identity.vpcEndpointId</code>	O ID do endpoint da VPC que está fazendo a solicitação ao endpoint do API Gateway. Presente somente quando você tem uma API privada.
<code>\$context.identity.user</code>	O identificador principal da entidade do usuário que será autorizado contra o acesso a recursos. Compatível com recursos que usam a autorização do IAM.
<code>\$context.identity.userAgent</code>	O cabeçalho User-Agent do autor da chamada da API.

Parâmetro	Descrição
<code>\$context.identity.userArn</code>	O Nome do Recurso Amazon (ARN) do usuário efetivo identificado após a autenticação. Para obter mais informações, consulte https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html .
<code>\$context.isCanaryRequest</code>	Retorna <code>true</code> se a solicitação foi direcionada ao canário e <code>false</code> se a solicitação não foi direcionada ao canário. Presente somente quando você tem um canário habilitado.
<code>\$context.path</code>	O caminho da solicitação. Por exemplo, para um URL de solicitação sem proxy de <code>https://{rest-api-id}.execute-api.{region}.amazonaws.com/{stage}/root/child</code> , o valor <code>\$context.path</code> é <code>/stage/root/child</code> .
<code>\$context.protocol</code>	O protocolo de solicitação, por exemplo, <code>HTTP/1.1</code> .

 Note

As APIs do API Gateway podem aceitar solicitações HTTP/2, mas o API Gateway envia solicitações para integrações de back-end usando HTTP/1.1. Como resultado, o protocolo de solicitação é registrado como HTTP/1.1 mesmo se um cliente enviar uma solicitação que usa HTTP/2.

Parâmetro	Descrição
<code>\$context.requestId</code>	Um ID para a solicitação. Os clientes podem substituir esse ID de solicitação. Usar <code>\$context.extendedRequestId</code> para um ID de solicitação exclusivo gerado pelo API Gateway.
<code>\$context.requestOverride.header. <i>header_name</i></code>	Substituição do cabeçalho da solicitação. Esse parâmetro, quando definido, contém os cabeçalhos a serem usados em lugar dos HTTP Headers (Cabeçalhos HTTP) que são definidos no painel Integration Request (Solicitação de integração). Para obter mais informações, consulte Substituir parâmetros de solicitação e resposta da API e códigos de status por APIs REST no API Gateway .
<code>\$context.requestOverride.path. <i>path_name</i></code>	Substituição do caminho da solicitação. Esse parâmetro, quando definido, contém o caminho da solicitação a ser usado em lugar dos URL Path Parameters (Parâmetros de caminho de URL) que são definidos no painel Integration Request (Solicitação de integração). Para obter mais informações, consulte Substituir parâmetros de solicitação e resposta da API e códigos de status por APIs REST no API Gateway .

Parâmetro	Descrição
<code>\$context.requestOverride.querystring. <i>querystring_name</i></code>	Substituição da string de consulta da solicitação. Esse parâmetro, quando definido, contém as strings de consulta da solicitação a serem usadas em lugar dos URL Query String Parameters (Parâmetros de string de consulta de URL) que são definidos no painel Integration Request (Solicitação de integração). Para obter mais informações, consulte Substituir parâmetros de solicitação e resposta da API e códigos de status por APIs REST no API Gateway .
<code>\$context.responseOverride.header. <i>header_name</i></code>	Substituição do cabeçalho da resposta. Esse parâmetro, quando definido, contém o cabeçalho a ser retornado em lugar do Response header (Cabeçalho de resposta) que é definido como o Default mapping (Mapeamento padrão) no painel Integration Response (Resposta de integração). Para obter mais informações, consulte Substituir parâmetros de solicitação e resposta da API e códigos de status por APIs REST no API Gateway .
<code>\$context.responseOverride.status</code>	Substituição do código de status da resposta. Esse parâmetro, quando definido, contém o código de status a ser retornado em lugar do Method response status (Status de resposta de método) que é definido como o Default mapping (Mapeamento padrão) no painel Integration Response (Resposta de integração). Para obter mais informações, consulte Substituir parâmetros de solicitação e resposta da API e códigos de status por APIs REST no API Gateway .

Parâmetro	Descrição
<code>\$context.requestTime</code>	O horário da solicitação CLF formatado (dd/MMM/yyyy:HH:mm:ss +-hhmm).
<code>\$context.requestTimeEpoch</code>	O tempo de solicitação formatado em Epoch , em milissegundos.
<code>\$context.resourceId</code>	O identificador que o API Gateway atribui ao seu recurso.
<code>\$context.resourcePath</code>	O caminho para o seu recurso. Por exemplo, para a URI de solicitação sem proxy de <code>https://{rest-api-id}.execute-api.{region}.amazonaws.com/{stage}/root/child</code> , o valor <code>\$context.resourcePath</code> é <code>/root/child</code> . Para obter mais informações, consulte Tutorial: Crie uma API REST com uma integração de não proxy de HTTP .
<code>\$context.stage</code>	O estágio de implantação da solicitação de API (por exemplo, Beta ou Prod).
<code>\$context.wafResponseCode</code>	A resposta recebida do AWS WAF : <code>WAF_ALLOW</code> ou <code>WAF_BLOCK</code> . Não será definido se o estágio não estiver associado a uma ACL da web. Para obter mais informações, consulte the section called “AWS WAF” .
<code>\$context.webaclArn</code>	O ARN completo da ACL da web que é utilizada para decidir se deseja permitir ou bloquear a solicitação. Não será definido se o estágio não estiver associado a uma ACL da web. Para obter mais informações, consulte the section called “AWS WAF” .

Variáveis de entrada

É possível usar as variáveis `$input` com distinção de maiúsculas e minúsculas a seguir para se referir à carga útil e aos parâmetros de solicitação de método. As seguintes funções estão disponíveis:

Variável e função	Descrição
<code>\$input.body</code>	Retorna a carga de solicitação bruta como uma string. É possível usar <code>\$input.body</code> para preservar números inteiros de ponto flutuante, como <code>10.00</code> .
<code>\$input.json(x)</code>	Essa função avalia uma expressão JSONPath e retorna os resultados como uma string JSON. Por exemplo, <code>\$input.json('\$.pets')</code> retorna uma string JSON que representa a estrutura <code>pets</code>. Para obter mais informações sobre o JSONPath, consulte JSONPath ou JSONPath para Java.
<code>\$input.params()</code>	Retorna um mapa de todos os parâmetros de solicitação. Recomendamos que você use <code>\$util.escapeJavaScript</code> para higienizar o resultado a fim de evitar um potencial ataque de injeção. Para o controle total da higienização de solicitações, use uma integração de proxy sem um modelo e realize a higienização de solicitações em sua integração.
<code>\$input.params(x)</code>	Retorna o valor de um parâmetro de solicitação do método do caminho, da string de consulta ou do valor de cabeçalho (pesquisados nessa ordem), considerando uma string de nome do parâmetro <code>x</code> . Recomendamos que você use <code>\$util.escapeJavaScript</code> para higienizar

Variável e função	Descrição
	remover o parâmetro a fim de evitar um potencial ataque de injeção. Para o controle total da higienização de parâmetros, use uma integração de proxy sem um modelo e realize a higienização de solicitações em sua integração.

Variável e função	Descrição
<code>\$input.path(x)</code>	Usa uma string de expressão JSONPath (x) e retorna uma representação de objeto JSON do resultado. Isso permite que você acesse e manipule elementos da carga nativamente em Apache Velocity Template Language (VTL). Por exemplo, se a expressão <code>\$input.path('\$.pets')</code> retorna um objeto da seguinte forma: <pre>[{ "id": 1, "type": "dog", "price": 249.99 }, { "id": 2, "type": "cat", "price": 124.99 }, { "id": 3, "type": "fish", "price": 0.99 }]</pre> <code>\$input.path('\$.pets').size()</code> retornaria "3". Para obter mais informações sobre o JSONPath, consulte JSONPath ou JSONPath para Java.

Variáveis de estágio

É possível usar as seguintes variáveis de estágio como espaços reservados para ARNs e URLs em integrações de método. Para obter mais informações, consulte [the section called “Usar variáveis de estágio”](#).

Sintaxe	Descrição
<code>\$stageVariables. <i>variable_name</i> ,</code> <code>\$stageVariables[' <i>variable_name</i> '],</code> ou <code>\${stageVariables[' <i>variable_name</i>']}</code>	<i>variable_name</i> representa um nome de variável de estágio.

Variáveis de utilidade

É possível usar as variáveis \$util com distinção de maiúsculas e minúsculas a seguir para utilizar funções utilitárias para associar modelos. A menos que especificado de outra forma, o conjunto de caracteres padrão é UTF-8.

Função	Descrição
<code>\$util.escapeJavaScript()</code>	Escapa os caracteres em uma string usando regras de string JavaScript.

 **Note**
Essa função transformará quaisquer aspas simples comuns (') em aspas com escape (\ '). No entanto, as aspas simples com escape não são válidas no JSON. Portanto, quando a saída dessa função for usada em uma propriedade JSON, você deverá transformar todas aspas simples (\ ') com escape de volta para aspas simples comuns ('). Isso é mostrado no exemplo a seguir:

Função	Descrição
	<pre>"input" : "\$util.escapeJavaScript(<i>data</i>).replaceAll("\\'", "'")"</pre>
<code>\$util.parseJson()</code>	Usa um JSON "transformado em string" e retorna uma representação de objeto do resultado. Você pode usar o resultado dessa função para acessar e manipular elementos da carga nativamente em Apache VTL (Velocity Template Language). Por exemplo, se tiver a seguinte carga: <pre>{"errorMessage":{"key1\":"var1\","key2\":{\"arr\":[1,2,3]}}}</pre> e usar o seguinte modelo de mapeamento <pre>#set (\$errorMessageObj = \$util.parseJson(\$input.path('\$errorMessage'))){ "errorMessageObjKey2ArrVal" : \$errorMessageObj.key2.arr[0] }</pre> Você receberá a seguinte saída: <pre>{ "errorMessageObjKey2ArrVal" : 1 }</pre>
<code>\$util.urlEncode()</code>	Converte uma string no formato "application/x-www-form-urlencoded".

Função	Descrição
<code>\$util.urlDecode()</code>	Decodifica uma string "application/x-www-form-urlencoded".
<code>\$util.base64Encode()</code>	Codifica os dados em uma string codificada em base64.
<code>\$util.base64Decode()</code>	Decodifica os dados de uma string codificada em base64.

Respostas do gateway para APIs REST no API Gateway

Uma resposta do gateway é identificada por um tipo de resposta definido pelo API Gateway. Essa resposta consiste em um código de status HTTP, um conjunto de cabeçalhos adicionais que são especificados por mapeamentos de parâmetros e uma carga útil que é gerada por um modelo de mapeamento não [VTL](#).

Na API REST do API Gateway, uma resposta de gateway é representada por [GatewayResponse](#). No OpenAPI, uma instância de GatewayResponse é descrita pela extensão [x-amazon-apigateway-gateway-responses.gatewayResponse](#).

Para permitir uma resposta de gateway, você configura uma resposta de gateway para um [tipo de resposta com suporte](#) em nível de API. Sempre que o API Gateway retorna uma resposta desse tipo, os mapeamentos de cabeçalho e os modelos de mapeamento de carga útil definidos na resposta de gateway são aplicados para retornar os resultados mapeados ao autor da chamada da API.

Na seção a seguir, mostramos como configurar respostas de gateway usando o console do API Gateway e a API REST do API Gateway.

Configurar respostas do gateway para personalizar respostas de erro

Se o API Gateway falhar ao processar uma solicitação de entrada, ele retornará ao cliente uma resposta de erro sem encaminhar essa solicitação ao backend de integração. Por padrão, a resposta de erro contém uma breve mensagem de erro descritiva. Por exemplo, se você tentar chamar uma operação em um recurso de API indefinido, receberá uma resposta de erro com a mensagem `{ "message": "Missing Authentication Token" }`. Se você não tem experiência com o API Gateway, talvez ache difícil compreender o que exatamente deu errado.

Para algumas das respostas de erro, o API Gateway permite personalização pelos desenvolvedores de APIs, de modo a retornar as respostas em diferentes formatos. Para o exemplo `Missing Authentication Token`, você pode adicionar uma dica à carga de resposta original com a causa possível, como neste exemplo: `{"message":"Missing Authentication Token", "hint":"The HTTP method or resources may not be supported."}`.

Quando a API faz a intermediação entre um intercâmbio externo e a Nuvem AWS, você usa modelos de mapeamento VTL para a solicitação de integração ou a resposta de integração para mapear a carga útil de um formato para outro. No entanto, os modelos de mapeamento VTL funcionam apenas para solicitações válidas com respostas bem-sucedidas.

Para solicitações inválidas, o API Gateway ignora completamente a integração e retorna uma resposta de erro. É necessário usar a personalização para renderizar as respostas de erro em um formato compatível com intercâmbio. Aqui, a personalização é renderizada em um modelo de mapeamento não VTL que oferece suporte apenas substituições de variáveis simples.

Generalizando a resposta de erro gerada pelo API Gateway para todas as respostas geradas pelo API Gateway, nós as chamamos de respostas de gateway. Isso distingue as respostas geradas pelo API Gateway das respostas de integração. Um modelo de mapeamento de resposta de gateway pode acessar valores de variáveis `$context` e valores de propriedades `$stageVariables`, bem como parâmetros de solicitação de método, no formato `method.request.param-position.param-name`.

Para obter mais informações sobre variáveis `$context`, consulte [Variáveis de contexto para transformações de dados](#). Para obter mais informações sobre o `$stageVariables`, consulte [Variáveis de estágio](#). Para obter mais informações sobre parâmetros de solicitação de método, consulte [the section called “Variáveis de entrada”](#).

Tópicos

- [Configurar uma resposta de gateway para uma API REST usando o console do API Gateway](#)
- [Configurar uma resposta de gateway usando a API REST do API Gateway](#)
- [Configurar a personalização da resposta de gateway no OpenAPI](#)
- [Tipos de resposta de gateway do API Gateway](#)

Configurar uma resposta de gateway para uma API REST usando o console do API Gateway

O exemplo a seguir mostra como configurar uma resposta de gateway para uma API REST usando o console do API Gateway

Para personalizar uma resposta de gateway usando o console do API Gateway

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST.
3. No painel de navegação principal, selecione Respostas do gateway.
4. Escolha um tipo de resposta e selecione Editar. Nesta demonstração, usamos Token de autenticação ausente como exemplo.
5. Você pode alterar o Código de status gerado pelo API Gateway para retornar um código de status diferente que atenda aos requisitos da API. Neste exemplo, a personalização altera o código de status de padrão (403) para 404 porque esta mensagem de erro ocorre quando um cliente chama um recurso inválido ou incompatível que pode ser considerado como não encontrado.
6. Para retornar os cabeçalhos personalizados, selecione Adicionar cabeçalho de resposta em Cabeçalhos de resposta. Para fins de ilustração, adicionamos os seguintes cabeçalhos 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
```

Nos mapeamentos de cabeçalho anteriores, um nome de domínio estático ('a.b.c') é mapeado para o cabeçalho Allow-Control-Allow-Origin, para permitir acesso do CORS à API, o cabeçalho da solicitação de entrada de x-amzn-RequestId é mapeado para request-id na resposta, a variável de caminho petId da solicitação de entrada é mapeada para o cabeçalho request-path na resposta e o parâmetro de consulta q da solicitação original é mapeado para o cabeçalho request-query da resposta.

7. Em Modelos de resposta, mantenha application/json para Tipo de conteúdo e insira o seguinte modelo de mapeamento de corpo no editor do Corpo do modelo:

```
{
```

```
"message": "$context.error.messageString",
"type": "$context.error.responseType",
"statusCode": "'404'",
"stage": "$context.stage",
"resourcePath": "$context.resourcePath",
"stageVariables.a": "$stageVariables.a"
}
```

Este exemplo mostra como mapear as propriedades `$context` e `$stageVariables` para propriedades do corpo de resposta de gateway.

8. Escolha Salvar alterações.
9. Implantar a API em um estágio novo ou existente.

Teste a resposta do gateway chamando o seguinte comando CURL, pressupondo que o URL de invocação do método da API correspondente seja `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 o parâmetro de string de consulta extra `q=1` não é compatível com a API, um erro é retornado da resposta do gateway especificado. Você receberá uma resposta do gateway semelhante à seguinte:

```
> 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-Mk93KgoxnfzDlTUh3f1mzA==

{
  "message": "Missing Authentication Token",
  "type": MISSING_AUTHENTICATION_TOKEN,
  "statusCode": '404',
  "stage": custErr,
  "resourcePath": /pets/{petId},
  "stageVariables.a": a
}

```

O exemplo anterior pressupõe que o backend da API seja [Pet Store](#) e que a API possua uma variável de estágio, a, definida.

Configurar uma resposta de gateway usando a API REST do API Gateway

Antes de personalizar uma resposta de gateway usando a API REST do API Gateway, você já deve ter criado uma API e obtido seu identificador. Para recuperar o identificador da API, você pode seguir a relação do link [restapi:gateway-responses](#) e examinar o resultado.

Para personalizar uma resposta de gateway usando a API REST do API Gateway

1. Para substituir uma instância de [GatewayResponse](#) inteira, chame a ação [gatewayresponse:put](#). Especifique um [responseType](#) desejado no parâmetro de caminho de URL e forneça na carga útil da solicitação os mapeamentos [statusCode](#), [responseParameters](#) e [responseTemplates](#).
2. Para atualizar parte de uma instância GatewayResponse, chame a ação [gatewayresponse:update](#). Especifique um responseType desejado no parâmetro de caminho de URL e forneça na carga útil da solicitação as propriedades individuais GatewayResponse desejadas, por exemplo, o mapeamento responseParameters ou responseTemplates.

Configurar a personalização da resposta de gateway no OpenAPI

Você pode usar a extensão x-amazon-apigateway-gateway-responses no nível raiz de API para personalizar respostas de gateway no OpenAPI. A seguinte definição do OpenAPI mostra um exemplo para personalizar a [GatewayResponse](#) do 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}"
    }
  }
}

```

Neste exemplo, a personalização altera o código de status do padrão (403) para 404. Ela também adiciona à resposta de gateway quatro parâmetros de cabeçalho e um modelo de mapeamento de corpo para o tipo de mídia `application/json`.

Tipos de resposta de gateway do API Gateway

O API Gateway expõe as seguintes respostas de gateway para personalização pelos desenvolvedores de APIs.

Tipo de resposta do Gateway	Código de status de padrão	Descrição
ACCESS_DENIED	403	A resposta de gateway para uma falha de autorização; por exemplo, quando o acesso é negado por um autorizador personalizado ou do Amazon Cognito. Se o tipo de resposta não for especificado, essa resposta assumirá como padrão o tipo <code>DEFAULT_4XX</code> .
API_CONFIGURATION_ERROR	500	A resposta do gateway para uma configuração de API inválida, incluindo quando um endereço de endpoint inválido

Tipo de resposta do Gateway	Código de status de padrão	Descrição
		é enviado, quando a decodificação em base64 falha nos dados binários quando o suporte binário está implementado ou quando o mapeamento da resposta de integração não pode corresponder a nenhum modelo e nenhum modelo padrão está configurado. Se o tipo de resposta não for especificado, essa resposta assumirá como padrão o tipo <code>DEFAULT_5XX</code> .
<code>AUTHORIZER_CONFIGURATION_ERROR</code>	<code>500</code>	A resposta de gateway por não conseguir se conectar a um autorizador personalizado ou do Amazon Cognito. Se o tipo de resposta não for especificado, essa resposta assumirá como padrão o tipo <code>DEFAULT_5XX</code> .
<code>AUTHORIZER_FAILURE</code>	<code>500</code>	A resposta de gateway quando um autorizador personalizado ou do Amazon Cognito falhou ao autenticar o autor da chamada. Se o tipo de resposta não for especificado, essa resposta assumirá como padrão o tipo <code>DEFAULT_5XX</code> .

Tipo de resposta do Gateway	Código de status de padrão	Descrição
BAD_REQUEST_PARAMETERS	400	A resposta de gateway quando o parâmetro de solicitação não pode ser validado de acordo com um validador de solicitação ativado. Se o tipo de resposta não for especificado, essa resposta assumirá como padrão o tipo DEFAULT_4XX .
BAD_REQUEST_BODY	400	A resposta de gateway quando o corpo da solicitação não pode ser validado de acordo com um validador de solicitação ativado. Se o tipo de resposta não for especificado, essa resposta assumirá como padrão o tipo DEFAULT_4XX .

Tipo de resposta do Gateway	Código de status de padrão	Descrição
DEFAULT_4XX	Nulo	A resposta de gateway padrão para um tipo de resposta não especificado com o código de status de 4XX. Alterar o código de status dessa resposta de gateway de retorno altera os códigos de status de todas as outras respostas 4XX para o novo valor. Redefinir esse código de status para nulo reverte os códigos de status de todas as outras respostas 4XX para seus valores originais.  Note As respostas personalizadas do AWS WAF têm precedência sobre as respostas personalizadas do gateway.

Tipo de resposta do Gateway	Código de status de padrão	Descrição
DEFAULT_5XX	Nulo	A resposta de gateway padrão para um tipo de resposta não especificado com um código de status de 5XX. Alterar o código de status dessa resposta de gateway de retorno altera os códigos de status de todas as outras respostas 5XX para o novo valor. Redefinir esse código de status para nulo reverte os códigos de status de todas as outras respostas 5XX para seus valores originais.
EXPIRED_TOKEN	403	A resposta de gateway para um erro de token de autenticação do AWS expirado. Se o tipo de resposta não for especificado, essa resposta assumirá como padrão o tipo DEFAULT_4XX .
INTEGRATION_FAILURE	504	A resposta de gateway para um erro de integração com falha. Se o tipo de resposta não for especificado, essa resposta assumirá como padrão o tipo DEFAULT_5XX .

Tipo de resposta do Gateway	Código de status de padrão	Descrição
INTEGRATION_TIMEOUT	504	A resposta de gateway para um erro de tempo limite de integração. Se o tipo de resposta não for especificado, essa resposta assumirá como padrão o tipo DEFAULT_5XX .
INVALID_API_KEY	403	A resposta do gateway para uma chave de API inválida enviada para um método que requer uma chave de API. Se o tipo de resposta não for especificado, essa resposta assumirá como padrão o tipo DEFAULT_4XX .
INVALID_SIGNATURE	403	A resposta do gateway para um erro de assinatura da AWS inválido. Se o tipo de resposta não for especificado, essa resposta assumirá como padrão o tipo DEFAULT_4XX .
MISSING_AUTHENTICATION_TOKEN	403	A resposta de gateway para um erro de token de autenticação ausente, incluindo casos quando o cliente tenta invocar um método de API ou recurso sem suporte. Se o tipo de resposta não for especificado, essa resposta assumirá como padrão o tipo DEFAULT_4XX .

Tipo de resposta do Gateway	Código de status de padrão	Descrição
QUOTA_EXCEEDED	429	A resposta de gateway para o erro de cota de plano de uso excedida. Se o tipo de resposta não for especificado, essa resposta assumirá como padrão o tipo DEFAULT_4XX .
REQUEST_TOO_LARGE	413	A resposta de gateway para o erro de solicitação muito grande. Se o tipo de resposta não for especificado, essa resposta assumirá como padrão: HTTP content length exceeded 10485760 bytes.
RESOURCE_NOT_FOUND	404	A resposta de gateway quando o API Gateway não consegue localizar o recurso especificado depois que uma solicitação de API passa na autenticação e na autorização, exceto para a autorização e a autenticação da chave de API. Se o tipo de resposta não for especificado, essa resposta assumirá como padrão o tipo DEFAULT_4XX .

Tipo de resposta do Gateway	Código de status de padrão	Descrição
THROTTLED	429	A resposta de gateway quando limites de controle de fluxo em nível de plano de uso, método, estágio ou conta são excedidos. Se o tipo de resposta não for especificado, essa resposta assumirá como padrão o tipo DEFAULT_4XX .
UNAUTHORIZED	401	A resposta de gateway quando o autorizador personalizado ou do Amazon Cognito falhou ao autenticar o autor da chamada.
UNSUPPORTED_MEDIA_TYPE	415	A resposta de gateway quando a carga é de um tipo de mídia sem suporte, se o comportamento de passagem direta estrita estiver habilitado. Se o tipo de resposta não for especificado, essa resposta assumirá como padrão o tipo DEFAULT_4XX .

Tipo de resposta do Gateway	Código de status de padrão	Descrição
WAF_FILTERED	403	A resposta de gateway quando uma solicitação é bloqueada pelo AWS WAF. Se o tipo de resposta não for especificado, essa resposta assumirá como padrão o tipo DEFAULT_4XX .

 **Note**
As [respostas personalizadas do AWS WAF](#) têm precedência sobre as respostas personalizadas do gateway.

CORS para APIs REST no API Gateway

O [compartilhamento de recursos entre origens \(CORS\)](#) é um recurso de segurança de navegador que restringe as solicitações HTTP entre origens que são iniciadas em scripts em execução no navegador. Consulte mais informações em [O que é CORS?](#).

Determinar se deseja habilitar o suporte ao CORS

Uma solicitação HTTP entre origens é uma solicitação que é feita para:

- Um domínio diferente (por exemplo, de `example.com` para `amazondomains.com`)
- Um subdomínio diferente (por exemplo, de `example.com` para `petstore.example.com`)
- Uma porta diferente (por exemplo, de `example.com` para `example.com:10777`)
- Um protocolo diferente (por exemplo, de `https://example.com` para `http://example.com`)

Se você não conseguir acessar sua API e receber uma mensagem de erro contendo `Cross-Origin Request Blocked`, talvez seja necessário habilitar o CORS.

As solicitações HTTP entre origens podem ser divididas em dois tipos: solicitações simples e solicitações não simples.

Habilitar o CORS para uma solicitação simples

Uma solicitação HTTP será simples se todas as condições a seguir forem verdadeiras:

- Ela é emitida em relação a um recurso de API que permite apenas solicitações GET, HEAD e POST.
- Se for uma solicitação de método POST, deverá incluir um cabeçalho `Origin`.
- O tipo de conteúdo da carga da solicitação é `text/plain`, `multipart/form-data` ou `application/x-www-form-urlencoded`.
- A solicitação não contém cabeçalhos personalizados.
- Quaisquer requisitos adicionais que estão listados na [Documentação do Mozilla CORS para solicitações simples](#).

Para solicitações simples do método POST de origem cruzada, a resposta de seu recurso precisa incluir o cabeçalho `Access-Control-Allow-Origin: '*'` ou `Access-Control-Allow-Origin: 'origin'`.

Todas as outras solicitações HTTP entre origens são solicitações não simples.

Habilitar o CORS para uma solicitação não simples

Se os recursos de sua API receberem solicitações não simples, você deverá habilitar o suporte CORS adicional, dependendo de seu tipo de integração.

Habilitar o CORS para integrações sem proxy

Para essas integrações, o [protocolo CORS](#) exige que o navegador envie uma solicitação de simulação ao servidor e aguarde a aprovação (ou uma solicitação de credenciais) do servidor antes de enviar a solicitação real. Você deve configurar sua API para enviar uma resposta apropriada à solicitação de simulação.

Como criar uma resposta de simulação:

1. Crie um método `OPTIONS` com uma integração de simulação.
2. Adicione os seguintes cabeçalhos de resposta à resposta do método 200:

- Access-Control-Allow-Headers
 - Access-Control-Allow-Methods
 - Access-Control-Allow-Origin
3. Defina o comportamento de passagem da integração como NEVER. Nesse caso, a solicitação de método de um tipo de conteúdo não mapeado será rejeitada com uma resposta HTTP 415 Tipo de mídia incompatível. Para obter mais informações, consulte [Comportamento de solicitação de método para cargas úteis sem modelos de mapeamento para APIs REST no API Gateway](#).
 4. Insira valores para os cabeçalhos de resposta. Para possibilitar todas as origens, todos os métodos e cabeçalhos comuns, use os seguintes valores de cabeçalho:
 - 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: '*'

Depois de criar a solicitação de simulação, você deve retornar o cabeçalho Access-Control-Allow-Origin: '*' ou Access-Control-Allow-Origin: '*origin*' para todos os métodos habilitados para CORS para pelo menos todas as 200 respostas.

Habilitar o CORS para integrações sem proxy usando o Console de gerenciamento da AWS

Você pode usar o Console de gerenciamento da AWS para habilitar o CORS. O API Gateway cria um método OPTIONS e tenta adiciona o cabeçalho Access-Control-Allow-Origin às respostas de integração de métodos existentes. Isso nem sempre funciona e, às vezes, você precisa modificar manualmente a resposta de integração para retornar o cabeçalho Access-Control-Allow-Origin de todos os métodos habilitados para CORS para pelo menos todas as 200 respostas.

Se você tiver tipos de mídia binários definidos como */* para a API, quando o API Gateway criar um método OPTIONS, altere `contentHandling` para `CONVERT_TO_TEXT`.

O comando [update-integration](#) abaixo altera `contentHandling` para `CONVERT_TO_TEXT` em uma solicitação de integração:

```
aws apigateway update-integration \  
  --rest-api-id abc123 \  
  --resource-id aaa111 \  
  --http-method OPTIONS \  
  --content-handling CONVERT_TO_TEXT
```

```
--patch-operations op='replace',path='/contentHandling',value='CONVERT_TO_TEXT'
```

O comando [update-integration-response](#) abaixo altera contentHandling para CONVERT_TO_TEXT em uma resposta de integração:

```
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 o suporte ao CORS para integrações de proxy

Para uma integração de proxy do Lambda ou integração de proxy HTTP, seu back-end é responsável por retornar os cabeçalhos Access-Control-Allow-Origin, Access-Control-Allow-Methods e Access-Control-Allow-Headers, porque uma integração de proxy não retorna uma resposta de integração.

As seguintes funções de exemplo do Lambda retornam os cabeçalhos de CORS necessários:

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!')
}
```

Tópicos

- [Ativar o CORS em um recurso usando o console do API Gateway](#)
- [Ativar o CORS em um recurso usando a API de importação do API Gateway](#)
- [Testar o CORS para uma API do API Gateway](#)

Ativar o CORS em um recurso usando o console do API Gateway

Você pode usar o console do API Gateway para habilitar o suporte ao CORS para um ou todos os métodos em um recurso de API REST que você criou. Depois de habilitar o suporte ao CORS, defina o comportamento de passagem da integração como NEVER. Nesse caso, a solicitação de método de um tipo de conteúdo não mapeado será rejeitada com uma resposta HTTP 415 Tipo de mídia incompatível. Para obter mais informações, consulte [Comportamento de solicitação de método para cargas úteis sem modelos de mapeamento para APIs REST no API Gateway](#)

Important

Os recursos podem conter recursos filho. A habilitação do suporte ao CORS para um recurso e seus métodos não o habilita recursivamente para recursos filho e seus métodos.

Para habilitar o suporte ao CORS em um recurso da API REST

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API.
3. Escolha um recurso em Recursos.
4. Na seção Detalhes do recurso, escolha Ativar CORS.

API Gateway > APIs > Resources - PetStore (abcd1234)

Resources

API actions ▼ Deploy API

Create resource

- /
 - GET
 - /pets
 - GET
 - OPTIONS
 - POST
 - /petId
 - GET
 - OPTIONS

Resource details

Path / Resource ID efg456

Update documentation Enable CORS

Methods (1)

Delete Create method

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

5. Na caixa Ativar CORS, faça o seguinte:

- (Opcional) Se você criou uma resposta de gateway personalizada e deseja habilitar o suporte ao CORS para uma resposta, selecione uma resposta de gateway.
- Selecione cada método para habilitar o suporte ao CORS. O método `OPTION` deve ter o CORS habilitado.

Se você habilitar o suporte ao CORS para um método `ANY`, o CORS será habilitado para todos os métodos.

- No campo de entrada `Access-Control-Allow-Headers`, insira uma string estática de uma lista separada por vírgulas de cabeçalhos que o cliente deve enviar na solicitação real do recurso. Use a lista de cabeçalhos `'Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token'` fornecida pelo console ou especifique seus próprios cabeçalhos.
- Use o valor `'*'` fornecido pelo console como o valor do cabeçalho `Access-Control-Allow-Origin` para permitir solicitações de acesso de todas as origens ou especifique uma origem que deve ter permissão para acessar o recurso.
- Escolha Salvar.

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

Enable CORS

CORS settings [Info](#)

To allow requests from scripts running in the browser, configure cross-origin resource sharing (CORS) for your API. When you save your configuration, API Gateway replaces any existing CORS settings with your new configuration.

Gateway responses

API Gateway will configure CORS for the selected gateway responses.

☐ Default 4XX

☐ Default 5XX

Access-Control-Allow-Methods

☐ GET

☒ OPTIONS

Access-Control-Allow-Headers

API Gateway will configure CORS for the selected gateway responses.

Content-Type,X-Amz-Date,Authorization,X-Api-Key,X-Amz-Security-Token

Access-Control-Allow-Origin

Enter an origin that can access the resource. Use a wildcard "*" to allow any origin to access the resource.

*

► Additional settings

[Cancel](#)

[Save](#)

Important

Ao aplicar as instruções acima ao método ANY em uma integração de proxy, nenhum dos cabeçalhos CORS aplicáveis será definido. Em vez disso, o backend deve retornar os cabeçalhos CORS aplicáveis, como `Access-Control-Allow-Origin`.

Depois que o CORS for habilitado no método GET, um método OPTIONS será adicionado ao recurso se ainda não estiver lá. A resposta 200 do método OPTIONS é automaticamente configurada para retornar os três cabeçalhos `Access-Control-Allow-*` a fim de atender a handshakes de simulação. Além disso, o método real (GET) também é configurado por padrão para retornar o cabeçalho `Access-Control-Allow-Origin` em sua resposta 200. Para outros tipos de respostas, você precisará configurá-las manualmente para retornar o cabeçalho `Access-Control-Allow-Origin` com "*" ou origens específicas, caso não queira retornar o erro `Cross-origin access`.

Depois que você habilitar o suporte ao CORS em seu recurso, você deve implantar ou reimplantar a API para que as novas configurações entrem em vigor. Para obter mais informações, consulte [the section called “Crie um implantação do .”](#).

Note

Se você não conseguir habilitar o suporte ao CORS em seu recurso depois de seguir o procedimento, recomendamos que você compare sua configuração de CORS com o recurso /pets da API de exemplo. Para saber como criar a API de exemplo, consulte [the section called “Tutorial: Criar uma API REST importando um exemplo”](#).

Ativar o CORS em um recurso usando a API de importação do API Gateway

Se você estiver usando o recurso [API Gateway Import API](#), poderá configurar o suporte para CORS usando um arquivo do OpenAPI. Primeiro, é necessário definir um método OPTIONS no seu recurso que retorne os cabeçalhos necessários.

Note

Os navegadores da Web esperam que cabeçalhos Access-Control-Allow-Headers e Access-Control-Allow-Origin sejam configurados em cada método de API que aceite solicitações CORS. Além disso, alguns navegadores primeiro fazem uma solicitação HTTP para um método OPTIONS no mesmo recurso e esperam receber os mesmos cabeçalhos.

Exemplo do método **Options**

O exemplo a seguir cria um método OPTIONS para uma integração 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"

```

Depois de configurar o método OPTIONS para o seu recurso, você poderá adicionar os cabeçalhos necessários aos outros métodos no mesmo recurso que precisam aceitar solicitações CORS.

1. Declare Access-Control-Allow-Origin e Headers para os tipos de resposta.

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. Na tag `x-amazon-apigateway-integration`, configure o mapeamento desses cabeçalhos para seus 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 : "'*'"
```

Exemplo de API

O exemplo a seguir cria uma API completa com um método OPTIONS e um método GET com uma integração de 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"

```

Testar o CORS para uma API do API Gateway

É possível testar a configuração de CORS da API invocando a API e verificando os cabeçalhos de CORS na resposta. O comando `curl` a seguir envia uma solicitação `OPTIONS` para uma API implantada.

```
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

```

Os cabeçalhos `Access-Control-Allow-Origin`, `Access-Control-Allow-Headers` e `Access-Control-Allow-Methods` na resposta mostram que a API oferece suporte ao CORS. Para obter mais informações, consulte [CORS para APIs REST no API Gateway](#).

Tipos de mídia binários para APIs REST no API Gateway

No API Gateway, a solicitação e a resposta da API têm uma carga de texto ou binária. Uma carga de texto é uma string JSON codificada por UTF-8. Uma carga binária é qualquer coisa além de uma carga de texto. A carga binária pode ser, por exemplo, um arquivo JPEG, um arquivo GZip ou um

arquivo XML. A configuração da API necessária para oferecer suporte a mídia binária depende do fato de a API usar ou não integrações de proxy ou não proxy.

Se você usa uma integração de proxy com streaming de resposta de carga útil, não precisa configurar seus tipos de mídia binária. Para obter mais informações, consulte [the section called “Transmitir a resposta de integração para suas integrações de proxy”](#).

AWS LambdaIntegrações de proxy do

Para lidar com cargas binárias para integrações de proxy do AWS Lambda, é necessário codificar com base64 a resposta da sua função. Você também deve configurar os [binaryMediaTypes](#) para sua API. A configuração `binaryMediaTypes` da sua API é uma lista de tipos de conteúdo que sua API trata como dados binários. São exemplos de tipos de mídia binária `image/png` ou `application/octet-stream`. É possível usar o caractere curinga (*) para cobrir vários tipos de mídia.

O API Gateway usa o primeiro cabeçalho `Accept` dos clientes para determinar se uma resposta deve retornar mídia binária. Para retornar mídia binária quando não for possível controlar a ordem dos valores de cabeçalho `Accept`, como solicitações de um navegador, defina os tipos de mídia binária da API como `*/*`.

Para obter um código de exemplo, consulte [the section called “Exibir mídia binária de uma integração do proxy do Lambda no API Gateway”](#).

Se você usa uma integração do proxy do Lambda com streaming de resposta de carga útil, não precisa configurar seus tipos de mídia binária. Para obter mais informações, consulte [the section called “Configurar uma integração de proxy do Lambda com streaming de resposta da carga útil”](#).

Integrações sem proxy

Para lidar com cargas binárias para integrações não proxy, adicione os tipos de mídia à lista [binaryMediaTypes](#) do recurso `RestApi`. A configuração `binaryMediaTypes` da sua API é uma lista de tipos de conteúdo que sua API trata como dados binários. Como alternativa, é possível definir as propriedades [contentHandling](#) nos recursos [Integration](#) e [IntegrationResponse](#). O valor `contentHandling` pode ser `CONVERT_TO_BINARY`, `CONVERT_TO_TEXT` ou indefinido.

Note

Em integrações `MOCK` ou privadas, não é possível usar a configuração das propriedades `contentHandling` no Console de gerenciamento da AWS. Você deve usar a AWS CLI, o CloudFormation ou um SDK para definir as propriedades `contentHandling`.

Dependendo do valor de `contentHandling` e de se o cabeçalho `Content-Type` da resposta ou o cabeçalho `Accept` da solicitação de entrada corresponder ou não a uma entrada na lista `binaryMediaTypes`, o API Gateway poderá codificar os bytes binários brutos como uma string codificada em base64, decodificar uma string codificada em base64 de volta aos seus bytes brutos ou transmitir o corpo sem modificação.

Você deve configurar a API da seguinte forma para dar suporte a cargas binárias para sua API no API Gateway:

- Adicione os tipos de mídia binários desejados à lista `binaryMediaTypes` no recurso [RestApi](#). Se essa propriedade e a propriedade `contentHandling` não estiverem definidas, as cargas serão tratadas como strings JSON codificadas em UTF-8.
- Trate a propriedade `contentHandling` do recurso [Integration](#).
 - Para que a carga de solicitação seja convertida de uma string codificada em base64 em seu blob binário, defina a propriedade como `CONVERT_TO_BINARY`.
 - Para que a carga de solicitação seja convertida de um blob binário em uma string codificada em base64, defina a propriedade como `CONVERT_TO_TEXT`.
 - Para transmitir a carga sem modificação, deixe a propriedade indefinida. Para transmitir uma carga binária sem modificação, também é necessário garantir que o `Content-Type` corresponda a uma das entradas `binaryMediaTypes` e que [os comportamentos de passagem](#) estejam habilitados para a API.
- Defina a propriedade `contentHandling` do recurso [IntegrationResponse](#). A propriedade `contentHandling`, o cabeçalho `Accept` nas solicitações do cliente e os `binaryMediaTypes` das APIs combinados determinam como o API Gateway lida com as conversões de tipo de conteúdo. Para obter mais detalhes, consulte [the section called “Conversões de tipo de conteúdo no API Gateway”](#).

Important

Quando uma solicitação contém vários tipos de mídia em seu cabeçalho `Accept`, o API Gateway respeita apenas o primeiro tipo de mídia `Accept`. Em situações em que não é possível controlar a ordem dos tipos de mídia `Accept` e o tipo de mídia do seu conteúdo binário não é o primeiro da lista, adicione o primeiro tipo de mídia `Accept` na lista `binaryMediaTypes` da sua API. O API Gateway lida com todos os tipos de conteúdo nesta lista como binários.

Por exemplo, para enviar um arquivo JPEG usando um elemento `<img>` em um navegador, o navegador pode enviar `Accept:image/webp,image/*,*/*;q=0.8` em uma solicitação. Ao adicionar `image/webp` à lista `binaryMediaTypes`, o endpoint recebe o arquivo JPEG como binário.

Para obter informações detalhadas sobre como o API Gateway lida com o texto e as cargas binárias, consulte [Conversões de tipo de conteúdo no API Gateway](#).

Conversões de tipo de conteúdo no API Gateway

A combinação dos `binaryMediaTypes` de sua API, os cabeçalhos em solicitações de clientes e a propriedade de integração `contentHandling` determinam como o API Gateway codifica as cargas.

A tabela a seguir mostra como o API Gateway converte a carga útil da solicitação para configurações específicas do cabeçalho `Content-Type` de uma solicitação, a lista `binaryMediaTypes` de um recurso [RestApi](#) e o valor da propriedade `contentHandling` do recurso [Integration](#).

Carga da solicitação de método	Cabeçalho Content-Type da solicitação	binaryMediaTypes	contentHandling	Carga da solicitação de integração
Dados de texto	Qualquer tipo de dados	Não definido	Não definido	String codificada em UTF8
Dados de texto	Qualquer tipo de dados	Não definido	CONVERT_TO_BINARY	Blob binário codificado em Base64
Dados de texto	Qualquer tipo de dados	Não definido	CONVERT_TO_TEXT	String codificada em UTF8
Dados de texto	Um tipo de dados de texto	Conjunto com tipos de mídia correspondentes	Não definido	Dados de texto

Carga da solicitação de método	Cabeçalho Content-Type da solicitação	binaryMediaTypes	contentHandling	Carga da solicitação de integração
Dados de texto	Um tipo de dados de texto	Conjunto com tipos de mídia correspondentes	CONVERT_TO_BINARY	Blob binário codificado em Base64
Dados de texto	Um tipo de dados de texto	Conjunto com tipos de mídia correspondentes	CONVERT_TO_TEXT	Dados de texto
Dados binários	Um tipo de dados binários	Conjunto com tipos de mídia correspondentes	Não definido	Dados binários
Dados binários	Um tipo de dados binários	Conjunto com tipos de mídia correspondentes	CONVERT_TO_BINARY	Dados binários
Dados binários	Um tipo de dados binários	Conjunto com tipos de mídia correspondentes	CONVERT_TO_TEXT	String codificada em Base64

A tabela a seguir mostra como o API Gateway converte a carga útil da resposta para configurações específicas do cabeçalho `Accept` de uma solicitação, a lista `binaryMediaTypes` de um recurso [RestApi](#) e o valor da propriedade `contentHandling` do recurso [IntegrationResponse](#).

Important

Quando uma solicitação contém vários tipos de mídia em seu cabeçalho `Accept`, o API Gateway respeita apenas o primeiro tipo de mídia `Accept`. Em situações em que não é possível controlar a ordem dos tipos de mídia `Accept` e o tipo de mídia do seu conteúdo binário não é o primeiro da lista, adicione o primeiro tipo de mídia `Accept` na lista `binaryMediaTypes` da sua API. O API Gateway lida com todos os tipos de conteúdo nesta lista como binários.

Por exemplo, para enviar um arquivo JPEG usando um elemento `<img>` em um navegador, o navegador pode enviar `Accept:image/webp,image/*,*/*;q=0.8` em uma solicitação. Ao adicionar `image/webp` à lista `binaryMediaTypes`, o endpoint recebe o arquivo JPEG como binário.

Carga da resposta de integração	Cabeçalho Accept da solicitação	binaryMediaTypes	contentHandling	Mapear a carga da resposta
Dados de texto ou binários	Um tipo de texto	Não definido	Não definido	String codificada em UTF8
Dados de texto ou binários	Um tipo de texto	Não definido	CONVERT_TO_BINARY	Blob codificado em Base64
Dados de texto ou binários	Um tipo de texto	Não definido	CONVERT_TO_TEXT	String codificada em UTF8
Dados de texto	Um tipo de texto	Conjunto com tipos de mídia correspondentes	Não definido	Dados de texto
Dados de texto	Um tipo de texto	Conjunto com tipos de mídia correspondentes	CONVERT_TO_BINARY	Blob codificado em Base64
Dados de texto	Um tipo de texto	Conjunto com tipos de mídia correspondentes	CONVERT_TO_TEXT	String codificada em UTF8
Dados de texto	Um tipo binário	Conjunto com tipos de mídia correspondentes	Não definido	Blob codificado em Base64
Dados de texto	Um tipo binário	Conjunto com tipos de mídia correspondentes	CONVERT_TO_BINARY	Blob codificado em Base64

Carga da resposta de integração	Cabeçalho Accept da solicitação	binaryMediaTypes	contentHandling	Mapear a carga da resposta
Dados de texto	Um tipo binário	Conjunto com tipos de mídia correspondentes	CONVERT_TO_TEXT	String codificada em UTF8
Dados binários	Um tipo de texto	Conjunto com tipos de mídia correspondentes	Não definido	String codificada em Base64
Dados binários	Um tipo de texto	Conjunto com tipos de mídia correspondentes	CONVERT_TO_BINARY	Dados binários
Dados binários	Um tipo de texto	Conjunto com tipos de mídia correspondentes	CONVERT_TO_TEXT	String codificada em Base64
Dados binários	Um tipo binário	Conjunto com tipos de mídia correspondentes	Não definido	Dados binários
Dados binários	Um tipo binário	Conjunto com tipos de mídia correspondentes	CONVERT_TO_BINARY	Dados binários
Dados binários	Um tipo binário	Conjunto com tipos de mídia correspondentes	CONVERT_TO_TEXT	String codificada em Base64

Ao converter uma carga de texto em um blob binário, o API Gateway pressupõe que os dados de texto sejam uma string decodificada em base64 e gera a saída dos dados binários como um blob decodificado em base64. Se a conversão falhar, ela retornará uma resposta 500, indicando um erro de configuração da API. Você não fornece um modelo de mapeamento para tal conversão, embora deva habilitar os [comportamentos de passagem direta](#) na API.

Ao converter uma carga binária em uma string de texto, o API Gateway sempre aplica uma codificação de base64 aos dados binários. Você pode definir um modelo de mapeamento para esse tipo de carga, mas pode acessar somente a string codificada em base64 no modelo de mapeamento por meio de `$input.body`, conforme mostrado no seguinte trecho de um exemplo de modelo de mapeamento.

```
{
  "data": "$input.body"
}
```

Para que a carga do binário seja transmitida sem modificação, você deve habilitar os [comportamentos de passagem direta](#) na API.

Ativação do suporte binário usando o console do API Gateway

Esta seção explica como habilitar o suporte a binários usando o console do API Gateway. Como exemplo, usamos uma API integrada ao Amazon S3. Nosso foco está nas tarefas para definir os tipos de mídia com suporte e especificar como a carga deve ser tratada. Para obter informações detalhadas sobre como criar uma API integrada do Amazon S3, consulte [Tutorial: Crie uma API REST como um proxy do Amazon S3](#).

Como habilitar o suporte binário usando o console do API Gateway

1. Definir tipos de mídia binários para a API:
 - a. Crie uma nova API ou escolha uma API existente. Para esse exemplo, definimos o nome da API como `FileMan`.
 - b. Na API selecionada no painel de navegação principal, selecione Configurações da API.
 - c. No painel Configurações da API, selecione Adicionar tipo de mídia binária na seção Tipos de mídia binária.
 - d. Selecione Adicionar tipo de mídia binária.
 - e. Insira um tipo de mídia necessário, por exemplo, **image/png**, no campo de texto de entrada. Se necessário, repita esta etapa para adicionar mais tipos de mídia. Para oferecer suporte a todos os tipos de mídia binários, especifique `/*/*`.
 - f. Escolha Salvar alterações.
2. Defina como as cargas de mensagem são manipuladas para o método de API:

- a. Crie um novo recuso ou escolha um recurso existente na API. Para este exemplo, usamos o recurso `/ {folder} / {item}`.
- b. Crie um novo método ou escolha um método existente no recurso. Como exemplo, usamos o método `GET / {folder} / {item}` integrado à ação `Object GET` no Amazon S3.
- c. Em Manuseio de conteúdo, selecione uma opção.

The screenshot shows the configuration interface for an API method. The 'Action type' section has 'Use path override' selected. The 'Path override' field contains `{bucket}/{object}`. The 'Execution role' field contains `arn:aws:iam::444455556666:role/s3-ApiGatewayS3ReadOnlyRole`. The 'Credential cache' dropdown is set to 'Do not add caller credentials to cache key'. The 'Content handling' section is highlighted with a red box and shows 'Passthrough' selected in the dropdown menu. A 'Learn more' link is visible next to the 'Content handling' header.

Escolha Passthrough (Passagem) se não desejar converter o corpo quando o cliente e o backend aceitarem o mesmo formato binário. Selecione Converter em texto para converter o corpo binário em uma string codificada em base64 quando, por exemplo, o back-end exigir que uma carga útil de solicitação binária seja transmitida como a propriedade JSON. E selecione Converter em binário quando o cliente enviar uma string codificada em base64, e o back-end exigir o formato binário original, ou quando o endpoint gerar uma string codificada em base64 e o cliente aceitar apenas a saída binária.

- d. Em Passagem do corpo da solicitação, selecione Quando não há modelos definidos (recomendado) para habilitar o comportamento de passagem no corpo da solicitação.

Você também pode selecionar Nunca. Isso significa que a API rejeitará dados com tipos de conteúdo que não tenham um modelo de mapeamento.

- e. Preserve o cabeçalho `Accept` da solicitação de entrada na solicitação de integração. Você deverá fazer isso se tiver definido `contentHandling` como `passthrough` e quiser substituir essa configuração em tempo de execução.

HTTP headers (2)			< 1 >
Name	Mapped from		Caching
Accept	method.request.header.Accept		Inactive
Content-Type	method.request.header.Content-Type		Inactive

- f. Para conversão em texto, defina um modelo de mapeamento para colocar os dados binários codificados em base64 no formato necessário.

Veja um exemplo de modelo de mapeamento para conversão em texto:

```
{
  "operation": "thumbnail",
  "base64Image": "$input.body"
}
```

O formato desse modelo de mapeamento depende das exigências de endpoint da entrada.

- g. Escolha Salvar.

Ativação do suporte binário usando a API REST do API Gateway

As tarefas a seguir mostram como habilitar o suporte binário usando chamadas de API REST do API Gateway.

Tópicos

- [Adicionar e atualizar tipos de mídia binária com suporte para uma API](#)
- [Configurar conversões de carga de solicitação](#)
- [Configurar conversões de carga de resposta](#)
- [Converter dados binários em dados de texto](#)
- [Converter dados de texto em uma carga binária](#)
- [Transmitir uma carga binária](#)

Adicionar e atualizar tipos de mídia binária com suporte para uma API

Para habilitar o API Gateway para oferecer suporte a um novo tipo de mídia binário, você deve adicionar o tipo de mídia binária à lista `binaryMediaTypes` do recurso `RestApi`. Por exemplo, para que o API Gateway lide com imagens JPEG, envie uma solicitação PATCH ao recurso `RestApi`:


```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/binaryMediaTypes/image~1jpeg"
  }
]
}
```

A especificação do tipo MIME de `image/jpeg` que faz parte do valor da propriedade `path` é escapada como `image~1jpeg`.

Para atualizar os tipos de mídia binária com suporte, substitua ou remova o tipo de mídia da lista `binaryMediaTypes` do recurso `RestApi`. Por exemplo, para alterar o suporte binário de arquivos JPEG para bytes brutos, envie uma solicitação PATCH para o recurso `RestApi`, da seguinte forma:

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [{
    "op" : "replace",
    "path" : "/binaryMediaTypes/image~1jpeg",
    "value" : "application/octet-stream"
  },
  {
    "op" : "remove",
    "path" : "/binaryMediaTypes/image~1jpeg"
  }]
}
```

Configurar conversões de carga de solicitação

Se o endpoint exigir uma entrada de binário, defina a propriedade `contentHandling` do recurso `Integration` como `CONVERT_TO_BINARY`. Para fazer isso, envie uma solicitação PATCH, da seguinte forma:

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/integration

{
  "patchOperations" : [ {
```

```
"op" : "replace",
"path" : "/contentHandling",
"value" : "CONVERT_TO_BINARY"
}]
}
```

Configurar conversões de carga de resposta

Se o cliente aceitar o resultado como um blob binário em vez de uma carga codificada em base64 retornada do endpoint, defina a propriedade `contentHandling` do recurso `IntegrationResponse` como `CONVERT_TO_BINARY`. Para fazer isso, envie uma solicitação `PATCH`, da seguinte forma:

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/integration/
responses/<status_code>

{
  "patchOperations" : [ {
    "op" : "replace",
    "path" : "/contentHandling",
    "value" : "CONVERT_TO_BINARY"
  } ]
}
```

Converter dados binários em dados de texto

Para enviar dados binários como uma propriedade JSON da entrada para o AWS Lambda ou Kinesis por meio do API Gateway, faça o seguinte:

1. Ative o suporte a cargas binárias da API adicionando o novo tipo de mídia binária de `application/octet-stream` à lista `binaryMediaTypes` da API.

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/binaryMediaTypes/application~1octet-stream"
  } ]
}
```

2. Defina `CONVERT_TO_TEXT` na propriedade `contentHandling` do recurso `Integration` e forneça um modelo de mapeamento para atribuir a string codificada em base64 dos dados binários a uma propriedade JSON. No exemplo a seguir, a propriedade JSON é `body` e `$input.body` mantém a string codificada em 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}\"}"
    }
  ]
}
```

Converter dados de texto em uma carga binária

Suponha que uma função do Lambda retorne um arquivo de imagem como uma string codificada em base64. Para transmitir essa saída binária para o cliente por meio do API Gateway, faça o seguinte:

1. Atualize a lista `binaryMediaTypes` da API adicionando o tipo de mídia binária de `application/octet-stream` se ele ainda não estiver na lista.

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/binaryMediaTypes/application~1octet-stream",
  }]
}
```

2. Defina a propriedade `contentHandling` do recurso `Integration` como `CONVERT_TO_BINARY`. Não defina um modelo de mapeamento. Quando você não define um modelo de mapeamento, o API Gateway chama o modelo de passagem para retornar o blob binário decodificado em base64 como o arquivo de imagem ao cliente.

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/
integration/responses/<status_code>

{
  "patchOperations" : [
    {
      "op" : "replace",
      "path" : "/contentHandling",
      "value" : "CONVERT_TO_BINARY"
    }
  ]
}
```

Transmitir uma carga binária

Para armazenar uma imagem em um bucket do Amazon S3 usando o API Gateway, faça o seguinte:

1. Atualize a lista `binaryMediaTypes` da API adicionando o tipo de mídia binária de `application/octet-stream`, se ele ainda não estiver na lista.

```
PATCH /restapis/<restapi_id>

{
  "patchOperations" : [ {
    "op" : "add",
    "path" : "/binaryMediaTypes/application~1octet-stream"
  }
]
```

2. Na propriedade `contentHandling` do recurso `Integration`, defina `CONVERT_TO_BINARY`. Defina `WHEN_NO_MATCH` como o valor da propriedade `passthroughBehavior` sem definir um modelo de mapeamento. Isso permite que o API Gateway invoque o modelo de passagem.

```
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"
    }
  ]
}
```

Importar e exportar codificações de conteúdo para o API Gateway

Para importar a lista `binaryMediaTypes` em uma [RestApi](#), use a seguinte extensão do API Gateway para o arquivo de definição do OpenAPI da API. A extensão também é usada para exportar as configurações de API.

- [Propriedade `x-amazon-apigateway-binary-media-types`](#)

Para importar e exportar o valor da propriedade `contentHandling` em recurso `Integration` ou `IntegrationResponse`, use as seguintes extensões do API Gateway para as definições do OpenAPI:

- [Objeto `x-amazon-apigateway-integration`](#)
- [Objeto `x-amazon-apigateway-integration.response`](#)

Exibir mídia binária de uma integração do proxy do Lambda no API Gateway

Para retornar mídia binária de uma [integração de proxy do AWS Lambda](#), codifique com base64 a resposta da sua função do Lambda. Também é necessário [configurar os tipos de mídia binária da API](#). Quando você configura os tipos de mídia binários da API, ela trata esse tipo de conteúdo como dados binários. O tamanho máximo da carga é 10 MB.

Note

Para usar um navegador da Web para invocar uma API com este exemplo de integração, defina os tipos de mídia binária da API como `*/*`. O API Gateway usa o primeiro cabeçalho `Accept` dos clientes para determinar se uma resposta deve retornar mídia binária. Para retornar mídia binária quando não for possível controlar a ordem dos valores de cabeçalho `Accept`, como solicitações de um navegador, defina os tipos de mídia binária da API como `*/*` (para todos os tipos de conteúdo).

O exemplo a seguir da função do Lambda pode retornar uma imagem binária do Amazon S3 ou texto para clientes. A resposta da função inclui um cabeçalho `Content-Type` para indicar ao cliente o tipo de dados que ele retorna. A função define de modo condicional a propriedade `isBase64Encoded` em sua resposta, dependendo do tipo de dados que ela retorna.

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 saber mais sobre tipos de mídia binária, consulte [Tipos de mídia binários para APIs REST no API Gateway](#).

Acessar arquivos binários no Amazon S3 por meio de uma API do API Gateway

Os exemplos a seguir mostram o arquivo do OpenAPI usado para acessar imagens no Amazon S3, como fazer download de uma imagem do Amazon S3 e como fazer upload de uma imagem para o Amazon S3.

Tópicos

- [Arquivo do OpenAPI de uma API de exemplo para acessar imagens no Amazon S3](#)
- [Fazer download de uma imagem do Amazon S3](#)
- [Carregar uma imagem para o Amazon S3](#)

Arquivo do OpenAPI de uma API de exemplo para acessar imagens no Amazon S3

O seguinte arquivo do OpenAPI demonstra uma API de amostra que ilustra o download de um arquivo de imagem do Amazon S3 e o upload de um arquivo de imagem para o Amazon S3. Essa API expõe os métodos GET `/s3?key={file-name}` e PUT `/s3?key={file-name}` para download e upload de um arquivo de imagem especificado. O método GET retorna o arquivo de imagem como uma string codificada em base64 como parte de uma saída JSON, seguindo o modelo de mapeamento fornecido, em uma resposta 200 OK. O método PUT obtém um blob binário bruto como entrada e retorna uma resposta 200 OK com uma carga útil vazia.

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"
  }
}
}

```

Fazer download de uma imagem do Amazon S3

Para fazer download de um arquivo de imagem (image.jpg) como um blob binário do 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

```

A resposta bem-sucedida tem a seguinte aparência:

```
200 OK HTTP/1.1
```

```
[raw bytes]
```

Os bytes brutos são retornados porque o cabeçalho `Accept` é definido como um tipo de mídia binário de `application/octet-stream`, e o suporte binário está habilitado para a API.

Como alternativa, para fazer download de um arquivo de imagem (image.jpg) como uma string codificada em base64 (formatada como uma propriedade JSON) no Amazon S3, adicione um modelo de resposta à resposta de integração 200, conforme mostrado no seguinte bloco de definição de OpenAPI em negrito:

```

"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}"

```

```
    }  
  }  
},
```

A solicitação para fazer download do arquivo de imagem é semelhante à seguinte:

```
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
```

A resposta bem-sucedida tem a seguinte aparência:

```
200 OK HTTP/1.1  
  
{  
  "image": "W3JhdyBieXRlc10="  
}
```

Carregar uma imagem para o Amazon S3

Para carregar um arquivo de imagem (image.jpg) como um blob binário no 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]
```

A resposta bem-sucedida tem a seguinte aparência:

```
200 OK HTTP/1.1
```

Para carregar um arquivo de imagem (image.jpg) como uma string codificada em base64 no 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=
```

A carga de entrada deve ser uma string codificada em base64, pois o valor do cabeçalho Content-Type está definido como application/json. A resposta bem-sucedida tem a seguinte aparência:

```
200 OK HTTP/1.1
```

Acessar arquivos binários no Lambda usando uma API do API Gateway

O exemplo de OpenAPI a seguir demonstra como acessar um arquivo binário no AWS Lambda por meio de uma API do API Gateway. Essa API expõe os métodos GET /lambda?key={file-name} e PUT /lambda?key={file-name} para download e upload de um arquivo de imagem especificado. O método GET retorna o arquivo de imagem como uma string codificada em base64 como parte de uma saída JSON, seguindo o modelo de mapeamento fornecido, em uma resposta 200 OK. O método PUT obtém um blob binário bruto como entrada e retorna uma resposta 200 OK com uma carga útil vazia.

Crie a função do Lambda que é chamada pela API, e ela deve retornar uma string codificada em base64 com application/json como cabeçalho Content-Type.

Tópicos

- [Arquivo do OpenAPI de uma API de exemplo para acessar imagens no Lambda](#)
- [Download de uma imagem do Lambda](#)
- [Carregar uma imagem para o Lambda](#)

Arquivo do OpenAPI de uma API de exemplo para acessar imagens no Lambda

O seguinte arquivo do OpenAPI demonstra uma API de exemplo que ilustra o download de um arquivo de imagem do Lambda e o upload de um arquivo de imagem para o Lambda.

OpenAPI 3.0

```
{
  "openapi": "3.0.0",
  "info": {
    "version": "2016-10-21T17:26:28Z",
    "title": "ApiName"
  },
```



```

"paths": {
  "/lambda": {
    "get": {
      "parameters": [
        {
          "name": "key",
          "in": "query",
          "required": false,
          "schema": {
            "type": "string"
          }
        }
      ],
      "responses": {
        "200": {
          "description": "200 response",
          "content": {
            "application/json": {
              "schema": {
                "$ref": "#/components/schemas/Empty"
              }
            }
          }
        },
        "500": {
          "description": "500 response"
        }
      },
      "x-amazon-apigateway-integration": {
        "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/arn:aws:lambda:us-east-1:123456789012:function:image/invocations",
        "type": "AWS",
        "credentials": "arn:aws:iam::123456789012:role/Lambda",
        "httpMethod": "POST",
        "requestTemplates": {
          "application/json": "{\n  \"imageKey\":\n    \"${input.params('key')}\"\n}"
        },
        "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')}\",\n    \"image\": \"\${input.body}\"}\n}"
        },
        "responses": {
            "default": {
                "statusCode": "500"
            },
            "2\\d{2}": {
                "statusCode": "200"
            }
        }
    }
}

},
"x-amazon-apigateway-binary-media-types": [
    "application/octet-stream",
    "image/jpeg"
],
"servers": [
    {
        "url": "https://abcdefghi.execute-api.us-east-1.amazonaws.com/{basePath}",
        "variables": {
            "basePath": {
                "default": "/v1"
            }
        }
    }
],
"components": {
    "schemas": {
        "Empty": {
            "type": "object",
            "title": "Empty Schema"
        }
    }
}
}

```

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "version": "2016-10-21T17:26:28Z",
    "title": "ApiName"
  },
  "host": "abcdefghi.execute-api.us-east-1.amazonaws.com",
  "basePath": "/v1",
  "schemes": [
    "https"
  ],
  "paths": {
    "/lambda": {
      "get": {
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "key",
            "in": "query",
            "required": false,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          },
          "500": {
            "description": "500 response"
          }
        },
        "x-amazon-apigateway-integration": {
          "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/arn:aws:lambda:us-east-1:123456789012:function:image/invocations",
          "type": "AWS",
          "credentials": "arn:aws:iam::123456789012:role/Lambda",
          "httpMethod": "POST",
```

```

    "requestTemplates": {
      "application/json": "{\n  \"imageKey\": \"${input.params('key')}\n}"
    },
    "responses": {
      "default": {
        "statusCode": "500"
      },
      "2\\d{2}": {
        "statusCode": "200",
        "responseTemplates": {
          "application/json": "{\n  \"image\": \"${input.body}\n}"
        }
      }
    }
  },
  "put": {
    "produces": [
      "application/json", "application/octet-stream"
    ],
    "parameters": [
      {
        "name": "key",
        "in": "query",
        "required": false,
        "type": "string"
      }
    ],
    "responses": {
      "200": {
        "description": "200 response",
        "schema": {
          "$ref": "#/definitions/Empty"
        }
      },
      "500": {
        "description": "500 response"
      }
    },
    "x-amazon-apigateway-integration": {
      "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/arn:aws:lambda:us-east-1:123456789012:function:image/invocations",
      "type": "AWS",
      "credentials": "arn:aws:iam::123456789012:role/Lambda",

```

```

    "httpMethod": "POST",
    "contentHandling" : "CONVERT_TO_TEXT",
    "requestTemplates": {
      "application/json": "{\n  \"imageKey\": \"${input.params('key')}\",\n  \"image\": \"${input.body}\"}\n}"
    },
    "responses": {
      "default": {
        "statusCode": "500"
      },
      "2\\d{2}": {
        "statusCode": "200"
      }
    }
  }
}
},
"x-amazon-apigateway-binary-media-types" : ["application/octet-stream", "image/jpeg"],
"definitions": {
  "Empty": {
    "type": "object",
    "title": "Empty Schema"
  }
}
}
}

```

Download de uma imagem do Lambda

Para baixar um arquivo de imagem (image.jpg) como um blob binário do 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

```

A resposta bem-sucedida tem a seguinte aparência:

```
200 OK HTTP/1.1
```

```
[raw bytes]
```

Para baixar um arquivo de imagem (image.jpg) como uma string codificada em base64, formatada como uma propriedade JSON, do 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
```

A resposta bem-sucedida tem a seguinte aparência:

```
200 OK HTTP/1.1

{
  "image": "W3JhdyBieXRlc10="
}
```

Carregar uma imagem para o Lambda

Para carregar um arquivo de imagem (image.jpg) como um blob binário no 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]
```

A resposta bem-sucedida tem a seguinte aparência:

```
200 OK
```

Para carregar um arquivo de imagem (image.jpg) como uma string codificada em base64 no 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=
```

A resposta bem-sucedida tem a seguinte aparência:

```
200 OK
```

Invocar APIs REST no API Gateway

Para chamar uma API implantada, os clientes enviam solicitações ao URL do serviço de componente do API Gateway para execução da API, conhecido como `execute-api`.

A URL base para APIs REST está no seguinte formato:

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

em que *api-id* é o identificador da API, *region* é a região da AWS e *stage* é o nome do estágio da implantação da API.

Important

Antes de invocar uma API, você deve implantá-la no API Gateway. Para encontrar instruções sobre como implantar uma API, consulte [Implantar APIs REST no API Gateway](#).

Tópicos

- [Obter o URL de invocação de uma API](#)
- [Invocação de uma API](#)
- [Use o console do API Gateway para testar um método de API REST](#)
- [Usar um SDK Java gerado pelo API Gateway para uma API REST](#)
- [Usar um SDK Android gerado pelo API Gateway para uma API REST](#)
- [Usar um SDK JavaScript gerado pelo API Gateway para uma API REST](#)
- [Usar um SDK Ruby gerado pelo API Gateway para uma API REST](#)
- [Usar o SDK iOS gerado pelo API Gateway para uma API REST em Objective-C ou Swift](#)

Obter o URL de invocação de uma API

Você pode usar o console, a AWS CLI ou uma definição de OpenAPI exportada para obter o URL de invocação de uma API.

Obter o URL de invocação de uma API usando o console

O procedimento a seguir mostra como obter o URL de invocação de uma API no console da API REST.

Obter o URL de invocação de uma API usando o console da API REST

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API implantada.
3. No painel de navegação principal, escolha Estágio.
4. Em Detalhes do estágio, escolha o ícone de cópia para copiar o URL de invocação da API.

Esse URL é para o recurso-raiz da sua API.

Stage details [Info](#) Edit

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

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

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

5. Para obter o URL de invocação de uma API para outro recurso na API, expanda o estágio no painel de navegação secundário e selecione um método.
6. Escolha o ícone de cópia para copiar o URL de invocação no nível de recurso da API.

Stages

Stage actions ▼

Create stage

prod

/

GET

/pets

GET

OPTIONS

POST

{petId}

GET

OPTIONS

Method overrides

Edit

By default, methods inherit stage-level settings. To customize settings for a method, configure method overrides.

This method inherits its settings from the 'prod' stage.

Invoke URL

https://abcd1234.execute-api.us-west-1.amazonaws.com/prod/pets

Obter o URL de invocação de uma API usando a AWS CLI

O procedimento a seguir mostra como obter o URL de invocação de uma API usando a AWS CLI.

Obter o URL de invocação de uma API usando a AWS CLI

1. Use o comando a seguir para obter `rest-api-id`. Esse comando retorna todos os valores `rest-api-id` em sua região. Para saber mais, consulte [get-rest-apis](#).

```
aws apigateway get-rest-apis
```

2. Substitua o exemplo `rest-api-id` pelo seu `rest-api-id`, substitua o exemplo `{stage-name}` pelo seu `{stage-name}` e substitua a `{region}` pela sua região.

```
https://{restapi_id}.execute-api.{region}.amazonaws.com/{stage_name}/
```

Obter o URL de invocação de uma API usando o arquivo de definição OpenAPI exportado da API

Também é possível construir esse URL-raiz combinando os campos `host` e `basePath` de um arquivo de definição OpenAPI exportado da API. Para encontrar instruções sobre como exportar sua API, consulte [the section called “Exportar uma API REST”](#).

Invocação de uma API

Você pode chamar sua API implantada usando o navegador, `curl` ou outras aplicações, como o [Postman](#).

Além disso, você pode usar o console do API Gateway para testar uma chamada de API. O teste usa o recurso `TestInvoke` do API Gateway, que permite testar a API antes de implantá-la. Para ter mais informações, consulte [the section called “Usar o console para testar um método de API REST”](#).

Note

Os valores de parâmetros de strings de consulta em uma URL de invocação não podem conter `%%`.

Invocar uma API usando um navegador da web

Se sua API permitir acesso anônimo, você poderá usar qualquer navegador da web para invocar qualquer método GET. Insira o URL de invocação completo na barra de endereço do navegador.

Para outros métodos ou chamadas que exigem autenticação, você deve especificar uma carga útil ou assinar as solicitações. É possível lidar com elas em um script por trás de uma página HTML ou em um aplicativo cliente usando um dos SDKs da AWS.

Invocar uma API usando `curl`

Você pode usar uma ferramenta como [curl](#) em seu terminal para chamar sua API. O exemplo de comando `curl` a seguir invoca o método GET no recurso `getUsers` do estágio `prod` de uma 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"
```

Use o console do API Gateway para testar um método de API REST

Use o console do API Gateway para testar um método de API REST.

Tópicos

- [Pré-requisitos](#)
- [Testar um método com o console do API Gateway](#)

Pré-requisitos

- Você deve especificar as configurações dos métodos que deseja testar. Siga as instruções em [Métodos para APIs REST no API Gateway](#).

Testar um método com o console do API Gateway

Important

Testar métodos com o console do API Gateway pode ocasionar alterações em recursos que não podem ser desfeitas. Testar um método com o console do API Gateway é o mesmo que chamar o método fora do console do API Gateway. Por exemplo, se você usar o console do API Gateway para chamar um método que exclui os recursos de uma API, se a chamada do método for bem-sucedida, os recursos da API serão excluídos.

Como testar um método

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST.

3. No painel Resources (Recursos), escolha o método que você deseja testar.
4. Selecione a guia Testar. Talvez seja necessário selecionar o botão de seta para a direita para mostrar a guia.

Method request

Integration request

Integration response

Method response

Test

Test method

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

Query strings

dog=2

Headers

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

header1:value1
header2:value2

Client certificate

No client certificates have been generated.

Test

Insira os valores em qualquer das caixas exibidas (como Strings de consulta, Cabeçalhos e Corpo da solicitação). O console inclui esses valores na solicitação de método no formulário json/aplicativo padrão.

Para ver opções adicionais que talvez você precise especificar, entre em contato com o proprietário da API.

5. Escolha Test (Testar). As informações a seguir serão exibidas:
 - Request (Solicitação) é o caminho do recurso que foi chamado para o método.
 - Status é o código de status HTTP da resposta.
 - Latência (ms) é o tempo entre a recepção da solicitação do chamador e a resposta retornada.
 - Corpo da resposta é o corpo de resposta HTTP.
 - Cabeçalhos de resposta são os cabeçalhos de resposta HTTP.

 Tip

Dependendo do mapeamento, o código de status HTTP, o corpo e os cabeçalhos de resposta podem ser diferentes dos enviados pela função do Lambda, pelo proxy HTTP ou pelo proxy de serviço da AWS.

- Os logs são as entradas simuladas do Amazon CloudWatch Logs que teriam sido gravadas se esse método fosse chamado fora do console do API Gateway.

 Note

Embora as entradas do CloudWatch Logs sejam simuladas, os resultados da chamada do método são reais.

Além de usar o console do API Gateway, você pode usar a AWS CLI ou um AWS SDK para o API Gateway testar a invocação de um método. Para fazer isso usando a AWS CLI, consulte [test-invoke-method](#).

Usar um SDK Java gerado pelo API Gateway para uma API REST

Nesta seção, descreveremos as etapas para usar um SDK Java gerado pelo API Gateway para uma API REST ao utilizar a API de [Calculadora simples](#) como exemplo. Antes de prosseguir, você deve concluir as etapas em [Gerar SDKs para APIs REST no API Gateway](#).

Para instalar e usar um SDK Java gerado pelo API Gateway

1. Extraia o conteúdo do arquivo .zip gerado pelo API Gateway que você baixou anteriormente.
2. Baixe e instale o [Apache Maven](#) (deve ser a versão 3.5 ou posterior).
3. Faça download e instale o [JDK 8](#).
4. Defina a variável de ambiente JAVA_HOME.
5. Acesse a pasta do SDK descompactada na qual o arquivo pom.xml está localizado. Essa pasta é generated-code por padrão. Execute o comando mvn install para instalar os arquivos de artefato compilados no seu repositório Maven local. Isso cria uma pasta target, que contém a biblioteca SDK compilada.
6. Digite o comando a seguir para criar um stub de projeto de cliente para chamar a API usando a biblioteca SDK instalada.

```
mvn -B archetype:generate \
  -DarchetypeGroupId=org.apache.maven.archetypes \
  -DgroupId=examples.aws.apig.simpleCalc.sdk.app \
  -DartifactId=SimpleCalc-sdkClient
```

Note

O separador \ no comando anterior é incluído para facilitar a leitura. O comando inteiro deve estar em uma única linha, sem separadores.

Esse comando cria um stub de aplicativo. O stub do aplicativo contém um arquivo `pom.xml` e uma pasta `src` no diretório raiz do projeto (`SimpleCalc-sdkClient` no comando anterior). Inicialmente, há dois arquivos de origem: `src/main/java/{package-path}/App.java` e `src/test/java/{package-path}/AppTest.java`. Neste exemplo, `{package-path}` é `examples/aws/apig/simpleCalc/sdk/app`. Este caminho de pacote é derivado do valor `DarchetypeGroupId`. Você pode usar o arquivo `App.java` como um modelo para o seu aplicativo cliente e pode adicionar outros na mesma pasta, se necessário. Você pode usar o arquivo `AppTest.java` como modelo de teste unitário para o seu aplicativo e pode adicionar outros arquivos de código de teste à mesma pasta de teste, conforme necessário.

7. Atualize as dependências de pacotes no arquivo `pom.xml` gerado para o seguinte, substituindo as propriedades `groupId`, `artifactId`, `version` e `name` do seu projeto, se necessário:

```
<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

Quando uma versão mais recente do artefato dependente de `aws-java-sdk-core` não for compatível com a versão especificada acima (1.11.94), você deve atualizar a tag `<version>` para a nova versão.

8. Em seguida, mostramos como chamar a API com o uso do SDK, chamando os métodos `getABOp(GetABOpRequest req)`, `getApiRoot(GetApiRootRequest req)` e `postApiRoot(PostApiRootRequest req)` do SDK. Esses métodos correspondem aos métodos GET `/a/b/op`, GET `?a=x&b=y&op=operator` e POST `/`, com uma carga de solicitações da API `{"a": x, "b": y, "op": "operator"}`, respectivamente.

Atualize o arquivo `App.java` da seguinte maneira:

```
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());

    }
}
```

No exemplo anterior, as definições de configuração usadas para instanciar o cliente SDK são para fins ilustrativos e não são necessariamente as melhores práticas recomendadas. Além disso, a chamada de `sdkClient.shutdown()` é opcional, especialmente se for necessário um controle preciso sobre quando liberar recursos.

Mostramos os padrões essenciais para chamar uma API usando um SDK do Java. É possível estender as instruções para chamar outros métodos de API.

Usar um SDK Android gerado pelo API Gateway para uma API REST

Nesta seção, descreveremos as etapas para usar um SDK Android gerado pelo API Gateway para uma API REST. Antes de avançar, você já deve ter completado as etapas em [Gerar SDKs para APIs REST no API Gateway](#).

Note

O SDK gerado não é compatível com o Android 4.4 e versões anteriores. Para ter mais informações, consulte [the section called “Observações importantes”](#).

Para instalar e usar um SDK Android gerado pelo API Gateway

1. Extraia o conteúdo do arquivo .zip gerado pelo API Gateway que você baixou anteriormente.
2. Baixe e instale o [Apache Maven](#) (de preferência a versão 3.x).
3. Faça download e instale o [JDK 8](#).
4. Defina a variável de ambiente `JAVA_HOME`.
5. Execute o comando `mvn install` para instalar os arquivos de artefato compilados no seu repositório Maven local. Isso cria uma pasta `target`, que contém a biblioteca SDK compilada.
6. Copie o arquivo do SDK (cujo nome é derivado do ID do Artifact e da versão do Artifact que você especificou ao gerar o SDK, por exemplo, `simple-calcsdk-1.0.0.jar`) da pasta `target`,

juntamente com todas as outras bibliotecas da pasta `target/lib`, para a pasta `lib` do seu projeto.

Se você usa o Android Studio, crie uma pasta `libs` no seu módulo de aplicativo cliente e copie o arquivo `.jar` necessário para essa pasta. Verifique se a seção de dependências no arquivo `gradle` do módulo contém o seguinte.

```
compile fileTree(include: ['*.jar'], dir: 'libs')
compile fileTree(include: ['*.jar'], dir: 'app/libs')
```

Certifique-se de que nenhum arquivo `.jar` duplicado seja declarado.

7. Use a classe `ApiClientFactory` para inicializar o SDK gerado pelo API Gateway. Por exemplo:

```
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 usar um provedor de credenciais do Amazon Cognito para autorizar chamadas para sua API, use a classe `ApiClientFactory` para transmitir um conjunto de credenciais da AWS usando o SDK gerado pelo API Gateway, como mostra o exemplo a seguir.

```
// 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 uma chave de API usando o SDK gerado pelo API Gateway, use um código semelhante ao seguinte.

```
ApiClientFactory factory = new ApiClientFactory()
```

```
.apiKey("YOUR_API_KEY");
```

Usar um SDK JavaScript gerado pelo API Gateway para uma API REST

O procedimento a seguir mostra como usar um SDK JavaScript gerado pelo API Gateway.

Note

Estas instruções supõem que você já concluiu as instruções em [Gerar SDKs para APIs REST no API Gateway](#).

Important

Se a API tiver apenas métodos ANY definidos, o pacote SDK gerado não conterá um arquivo `apigClient.js`, e você precisará definir os métodos ANY por conta própria.

Para instalar, inicie e chame um SDK JavaScript gerado pelo API Gateway para uma API REST

1. Extraia o conteúdo do arquivo .zip gerado pelo API Gateway que você baixou anteriormente.
2. Habilite o CORS (compartilhamento de recursos entre origens) para todos os métodos que o SDK gerado pelo API Gateway chamará. Para obter instruções, consulte [CORS para APIs REST no API Gateway](#).
3. Na sua página da Web, inclua referências aos seguintes 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. No seu código, inicialize o SDK gerado pelo API Gateway usando um código semelhante ao seguinte.

```
var apigClient = apigClientFactory.newClient();
```

Para inicializar o SDK gerado pelo API Gateway com credenciais da AWS, use um código semelhante ao seguinte. Se você usar credenciais da AWS, todas as solicitações para a API serão assinadas.

```
var apigClient = apigClientFactory.newClient({
  accessKey: 'ACCESS_KEY',
  secretKey: 'SECRET_KEY',
});
```

Para usar uma chave de API com o SDK gerado pelo API Gateway, transmita essa chave de API como um parâmetro ao objeto Factory, usando um código semelhante ao seguinte. Se você usar uma chave de API, ela será especificada como parte do cabeçalho `x-api-key`, e todas as solicitações para essa API serão assinadas. Isso significa que você deve definir os cabeçalhos Accept CORS apropriados para cada solicitação.

```
var apigClient = apigClientFactory.newClient({
  apiKey: 'API_KEY'
});
```

5. Chame os métodos de API no API Gateway usando um código semelhante ao seguinte. Cada chamada retorna uma promessa com retornos de chamada de êxito e falha.

```
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.
  });

```

Aqui, *methodName* é construído a partir do caminho de recurso da solicitação de método e do verbo HTTP. Para a API SimpleCalc, os métodos de SDK de métodos referentes aos métodos de API de

1. GET /?a=...&b=...&op=...
2. POST /
- ```
{ "a": ..., "b": ..., "op": ...}
```
3. GET /{a}/{b}/{op}

métodos SDK correspondentes são os seguintes:

1. rootGet(params); // where params={"a": ..., "b": ..., "op": ...} is resolved to the query parameters
2. rootPost(null, body); // where body={"a": ..., "b": ..., "op": ...}
3. aB0pGet(params); // where params={"a": ..., "b": ..., "op": ...} is resolved to the path parameters



## Usar um SDK Ruby gerado pelo API Gateway para uma API REST

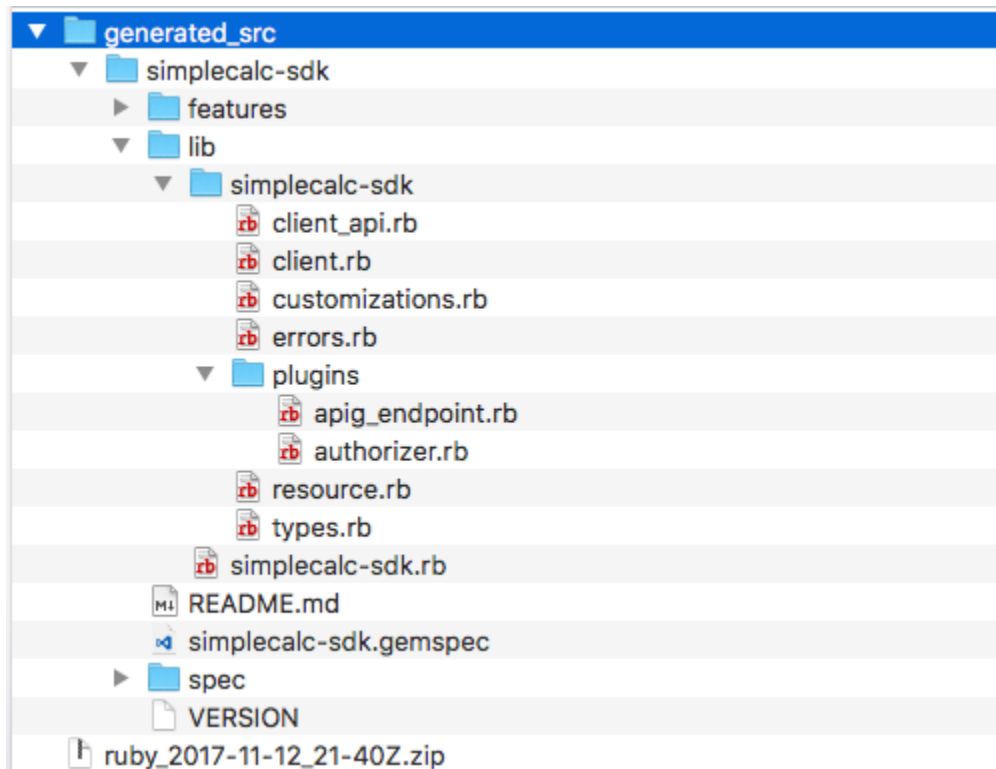
O procedimento a seguir mostra como usar um SDK Ruby gerado pelo API Gateway.

### Note

Estas instruções supõem que você já tenha concluído as instruções em [Gerar SDKs para APIs REST no API Gateway](#).

Para instalar, instancie e chame um SDK Ruby gerado pelo API Gateway para uma API REST

1. Descompacte o arquivo SDK Ruby baixado. A origem do SDK gerado é mostrada da seguinte forma.



2. Crie um Ruby Gem a partir da origem do SDK gerado, usando os seguintes comandos shell em uma janela de terminal:

```
change to /simplecalc-sdk directory
cd simplecalc-sdk

build the generated gem
```

```
gem build simplecalc-sdk.gemspec
```

Depois disso, simplecalc-sdk-1.0.0.gem se torna disponível.

3. Instale o gem:

```
gem install simplecalc-sdk-1.0.0.gem
```

4. Crie um aplicativo de cliente. Instancie e inicialize o cliente de SDK Ruby no aplicativo:

```
require 'simplecalc-sdk'
client = SimpleCalc::Client.new(
 http_wire_trace: true,
 retry_limit: 5,
 http_read_timeout: 50
)
```

Se a API com autorização do tipo AWS\_IAM estiver configurada, você pode incluir as credenciais da AWS do autor da chamada, fornecendo `accessKey` e `secretKey` durante a inicialização:

```
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. Faça chamadas de API por meio do SDK no aplicativo.

 Tip

Se você não estiver familiarizado com as convenções de chamada do método do SDK, você pode verificar o arquivo `client.rb` na pasta `lib` do SDK gerado. A pasta contém a documentação de cada chamada de método de API suportada.

Para descobrir as operações suportadas:

```
to show supported operations:
```

```
puts client.operation_names
```

Isso resulta na seguinte tela, correspondente aos métodos de API de GET /?

a={.}&b={.}&op={.}, GET /{a}/{b}/{op} e POST /, além de uma carga no formato {a:"...", b:"...", op:"..."}, respectivamente:

```
[:get_api_root, :get_ab_op, :post_api_root]
```

Para invocar o método de API de GET /?a=1&b=2&op=+, chame o método de SDK Ruby a seguir:

```
resp = client.get_api_root({a:"1", b:"2", op:"+"})
```

Para invocar o método de API de POST / com uma carga de {a: "1", b: "2", "op": "+"}, chame o método de SDK Ruby a seguir:

```
resp = client.post_api_root(input: {a:"1", b:"2", op:"+"})
```

Para invocar o método de API de GET /1/2/+, chame o método de SDK Ruby a seguir:

```
resp = client.get_ab_op({a:"1", b:"2", op:"+"})
```

As chamadas bem-sucedidas do método de SDK retornam a seguinte resposta:

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

## Usar o SDK iOS gerado pelo API Gateway para uma API REST em Objective-C ou Swift

Neste tutorial, mostraremos como usar um SDK do iOS gerado pelo API Gateway para uma API REST em um aplicativo Objective-C ou Swift para chamar a API subjacente. Usaremos a [API SimpleCalc](#) como exemplo para ilustrar os tópicos a seguir:

- Como instalar os componentes necessários do SDK Móvel da AWS no seu projeto Xcode
- Como criar o objeto de cliente da API antes de chamar os métodos da API
- Como chamar os métodos de API por meio dos métodos do SDK correspondentes no objeto de cliente da API
- Como preparar uma entrada de método e analisar seu resultado usando as classes de modelo correspondentes do SDK

### Tópicos

- [Usar o SDK do iOS \(Objective-C\) gerado para chamar a API](#)
- [Usar o SDK do iOS \(Swift\) gerado para chamar a API](#)

### Usar o SDK do iOS (Objective-C) gerado para chamar a API

Antes de iniciar o procedimento a seguir, você deve concluir as etapas em [Gerar SDKs para APIs REST no API Gateway](#) para o iOS no Objective-C e baixar o arquivo .zip do SDK gerado.

Instalar o SDK móvel da AWS e um SDK do iOS gerado pelo API Gateway em um projeto Objective-C

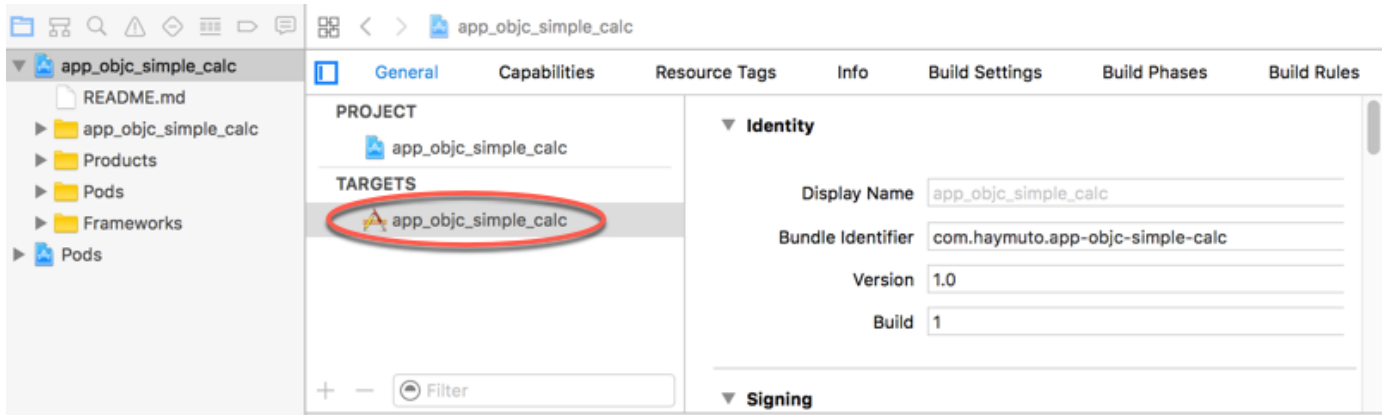
O procedimento a seguir descreve como instalar o SDK.

Para instalar e usar um SDK do iOS gerado pelo API Gateway no Objective-C

1. Extraia o conteúdo do arquivo .zip gerado pelo API Gateway que você baixou anteriormente. Usando a [API SimpleCalc](#), você pode querer renomear a pasta do SDK descompactada para algo como **sdk\_objc\_simple\_calc**. Nesta pasta do SDK, há um arquivo README .md file e um arquivo Podfile. O arquivo README .md contém as instruções para instalar e usar o SDK. Este tutorial fornece detalhes sobre essas instruções. A instalação utiliza o [CocoaPods](#) para importar as bibliotecas necessárias do API Gateway e outros componentes dependentes do SDK móvel da AWS. Você deve atualizar o Podfile para importar os SDKs para o projeto

Xcode rápida do seu aplicativo. A pasta do SDK não arquivada também contém uma pasta `generated-src`, que contém o código-fonte do SDK gerado da sua API.

2. Inicie o Xcode e crie um novo projeto Objective-C do iOS. Anote o destino do projeto. Você precisará defini-lo no Podfile.



3. Para importar o AWS Mobile SDK for iOS no projeto Xcode usando o CocoaPods, faça o seguinte:

- a. Instale o CocoaPods executando o seguinte comando em uma janela de terminal:

```
sudo gem install cocoapods
pod setup
```

- b. Copie o arquivo Podfile da pasta do SDK extraído no mesmo diretório que contém seu arquivo de projeto Xcode. Substitua o seguinte bloco:

```
target '<YourXcodeTarget>' do
 pod 'AWSAPIGateway', '~> 2.4.7'
end
```

com o nome de destino do seu projeto:

```
target 'app_objc_simple_calc' do
 pod 'AWSAPIGateway', '~> 2.4.7'
end
```

Se o seu projeto Xcode já contiver um arquivo chamado Podfile, adicione a seguinte linha de código a ele:

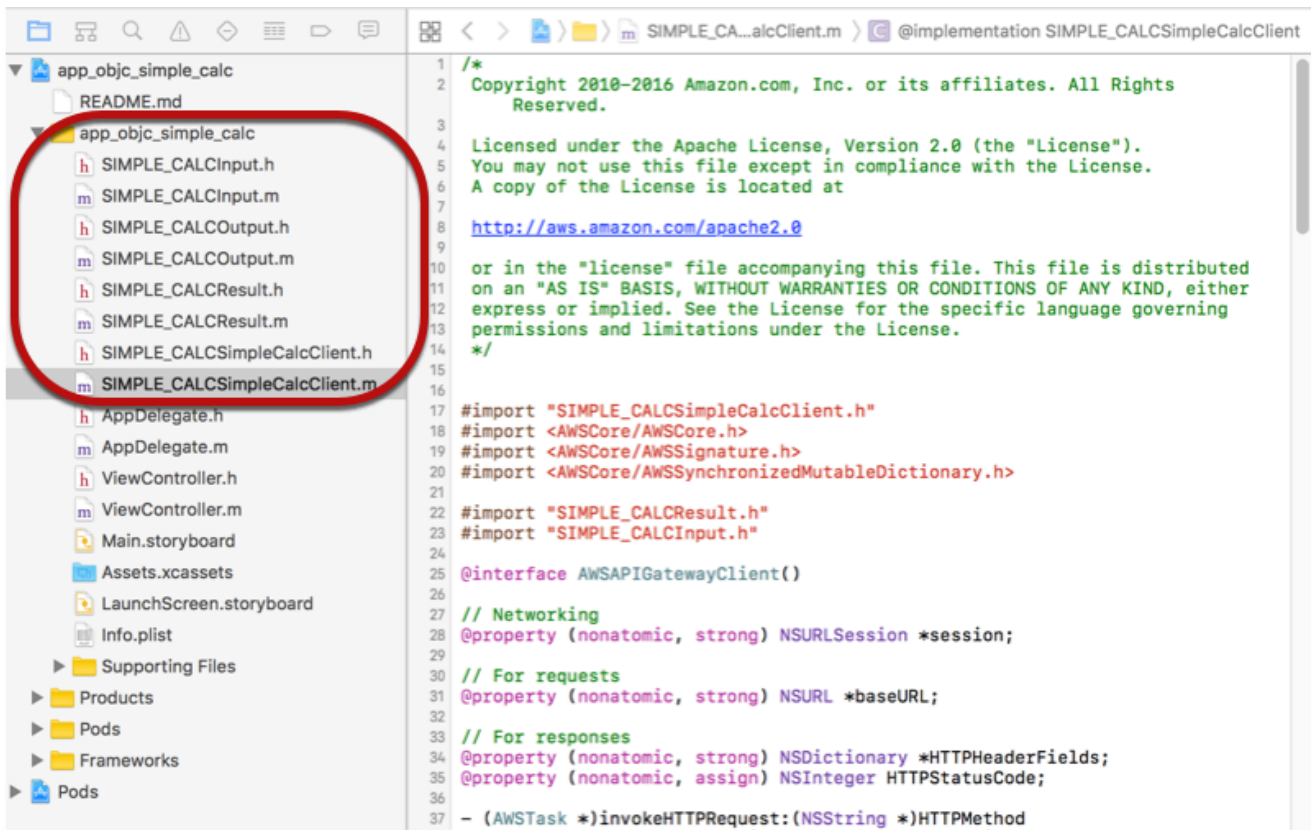
```
pod 'AWSAPIGateway', '~> 2.4.7'
```

- c. Abra uma janela de terminal e execute o seguinte comando:

```
pod install
```

Isso instala o componente do API Gateway e outros componentes dependentes do SDK móvel da AWS.

- d. Feche o projeto Xcode e abra o arquivo `.xcworkspace` para reiniciar o Xcode.
- e. Adicione todos os arquivos `.h` e `.m` do diretório `generated-src` do SDK extraído ao seu projeto Xcode.



Para importar o AWS Mobile SDK for iOS Objective-C no seu projeto, fazendo download explicitamente do SDK Móvel da AWS ou usando o [Carthage](#), siga as instruções no arquivo `README.md`. Certifique-se de usar apenas uma dessas opções para importar o SDK Móvel da AWS.

Chamar métodos de API usando o SDK do iOS gerado pelo API Gateway em um projeto Objective-C

Quando você gerou o SDK com o prefixo de SIMPLE\_CALC para essa API [SimpleCalc](#) com dois modelos para a entrada (Input) e a saída (Result) dos métodos, no SDK, a classe de cliente de API resultante torna-se SIMPLE\_CALCSimpleCalcClient e as classes de dados correspondentes são SIMPLE\_CALCInput e SIMPLE\_CALCResult, respectivamente. As solicitações e respostas da API são mapeadas para os métodos do SDK, da seguinte maneira:

- A solicitação de API de

```
GET /?a=...&b=...&op=...
```

torna-se o método SDK de

```
(AWSTask *)rootGet:(NSString *)op a:(NSString *)a b:(NSString *)b
```

A propriedade `AWSTask.result` é do tipo `SIMPLE_CALCResult`, se o modelo `Result` foi adicionado à resposta do método. Caso contrário, a propriedade será do tipo `NSDictionary`.

- Essa solicitação de API de

```
POST /

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

torna-se o método SDK de

```
(AWSTask *)rootPost:(SIMPLE_CALCInput *)body
```

- A solicitação de API de

```
GET /{a}/{b}/{op}
```

torna-se o método SDK de

```
(AWSTask *)aBOPGet:(NSString *)a b:(NSString *)b op:(NSString *)op
```

O procedimento a seguir descreve como chamar os métodos de API no código-fonte do aplicativo Objective-C; por exemplo, como parte do `viewDidLoad` delegado em um arquivo `ViewController.m`.

Como chamar a API por meio do SDK do iOS gerado pelo API Gateway

1. Importe o arquivo de cabeçalho da classe de cliente da API para tornar essa classe chamável no aplicativo:

```
#import "SIMPLE_CALCSimpleCalc.h"
```

A instrução `#import` também importa `SIMPLE_CALCInput.h` e `SIMPLE_CALCResult.h` para as duas classes de modelo.

2. Instancie a classe de cliente da API:

```
SIMPLE_CALCSimpleCalcClient *apiInstance = [SIMPLE_CALCSimpleCalcClient
 defaultClient];
```

Para usar o Amazon Cognito com a API, defina a propriedade `defaultServiceConfiguration` no objeto `AWSServiceManager` padrão, conforme mostrado a seguir, antes de chamar o método `defaultClient` para criar o objeto de cliente da API (mostrado no exemplo 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. Chame o método GET `/?a=1&b=2&op=+` para realizar 1+2:

```
[[apiInstance rootGet: @"+" a:@"1" b:@"2"] continueWithBlock:^id _Nullable(AWSTask
 * _Nonnull task) {
 _textField1.text = [self handleApiResponse:task];
```



```
 return nil;
}];
```

em que a função auxiliar `handleApiResponse:task` formata o resultado como uma string a ser exibida em um 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:@"%d %d %d = %d\n", task.result.input.a,
task.result.input.op, task.result.input.b, task.result.output.c];
 }
 return nil;
}
```

A exibição resultante é  $1 + 2 = 3$ .

#### 4. Chame a carga POST / para realizar 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;
}];
```

A exibição resultante é  $1 - 2 = -1$ .

#### 5. Chame GET /{a}/{b}/{op} para realizar 1/2:

```
[[apiInstance aB0pGet:@"1" b:@"2" op:@"div"] continueWithBlock:^id
_Nonnull(AWSTask * _Nonnull task) {
 _textField3.text = [self handleApiResponse:task];
 return nil;
}];
```

A exibição resultante é  $1 \div 2 = 0.5$ . Aqui, `div` é usado no lugar de `/` porque a [função do Lambda simples](#) no backend não manuseia variáveis de caminho codificadas por URL.

Usar o SDK do iOS (Swift) gerado para chamar a API

Antes de iniciar o procedimento a seguir, você deve concluir as etapas em [Gerar SDKs para APIs REST no API Gateway](#) para o iOS no Swift e baixar o arquivo .zip do SDK gerado.

## Tópicos

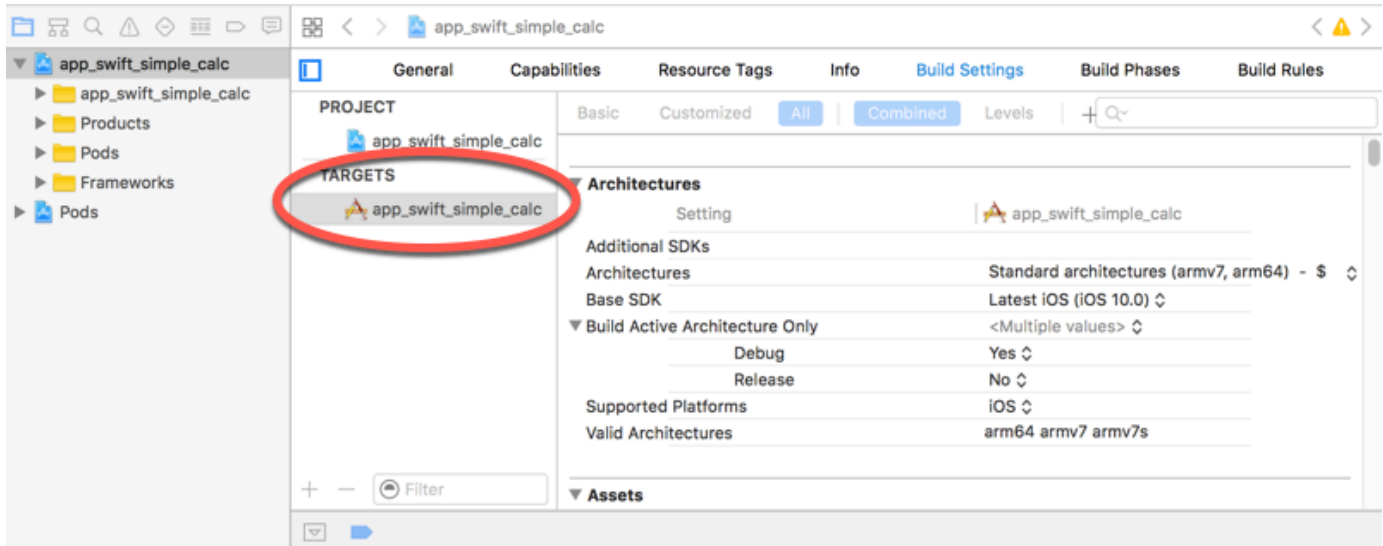
- [Instalar o SDK móvel da AWS e o SDK gerado pelo API Gateway em um projeto Swift](#)
- [Chamar métodos de API por meio do SDK do iOS gerado pelo API Gateway em um projeto Swift](#)

Instalar o SDK móvel da AWS e o SDK gerado pelo API Gateway em um projeto Swift

O procedimento a seguir descreve como instalar o SDK.

Para instalar e usar um SDK do iOS gerado pelo API Gateway no Swift

1. Extraia o conteúdo do arquivo .zip gerado pelo API Gateway que você baixou anteriormente. Usando a [API SimpleCalc](#), você pode querer renomear a pasta do SDK descompactada para algo como **sdk\_swift\_simple\_calc**. Nesta pasta do SDK, há um arquivo `README.md` file e um arquivo `Podfile`. O arquivo `README.md` contém as instruções para instalar e usar o SDK. Este tutorial fornece detalhes sobre essas instruções. A instalação utiliza o [CocoaPods](#) para importar os componentes necessários do SDK Móvel da AWS. Você deve atualizar o `Podfile` para importar os SDKs para o projeto Xcode rápida do seu aplicativo Swift. A pasta do SDK não arquivada também contém uma pasta `generated-src`, que contém o código-fonte do SDK gerado da sua API.
2. Inicie o Xcode e crie um novo projeto Swift do iOS. Anote o destino do projeto. Você precisará defini-lo no `Podfile`.



3. Para importar os componentes necessários do SDK Móvel da AWS no projeto Xcode usando o CocoaPods, faça o seguinte:

- a. Se o CocoaPods não estiver instalado, instale-o executando o seguinte comando em uma janela de terminal:

```
sudo gem install cocoapods
pod setup
```

- b. Copie o arquivo Podfile da pasta do SDK extraído no mesmo diretório que contém seu arquivo de projeto Xcode. Substitua o seguinte bloco:

```
target '<YourXcodeTarget>' do
 pod 'AWSAPIGateway', '~> 2.4.7'
end
```

pelo nome de destino do seu projeto, conforme mostrado:

```
target 'app_swift_simple_calc' do
 pod 'AWSAPIGateway', '~> 2.4.7'
end
```

Se o seu projeto Xcode já contiver um Podfile com o destino correto, basta adicionar a seguinte linha de código ao loop do `... end`:

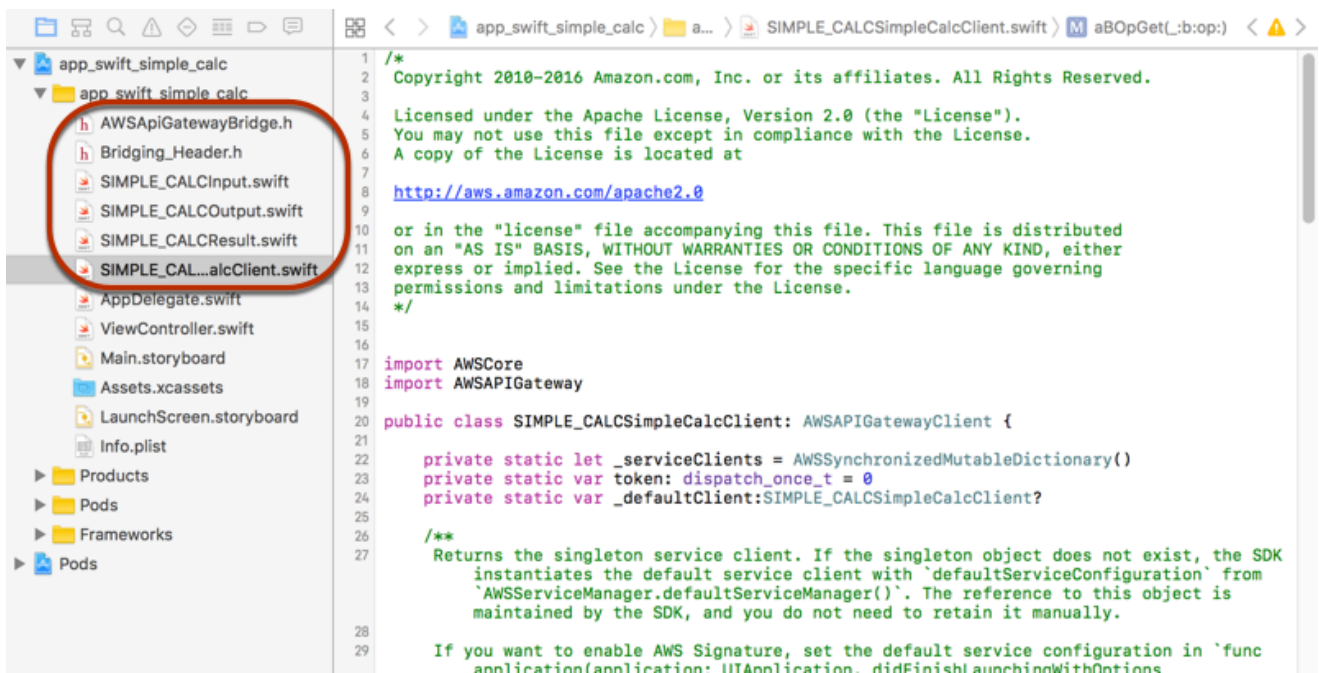
```
pod 'AWSAPIGateway', '~> 2.4.7'
```

- c. Abra uma janela de terminal e execute o seguinte comando no diretório do aplicativo:

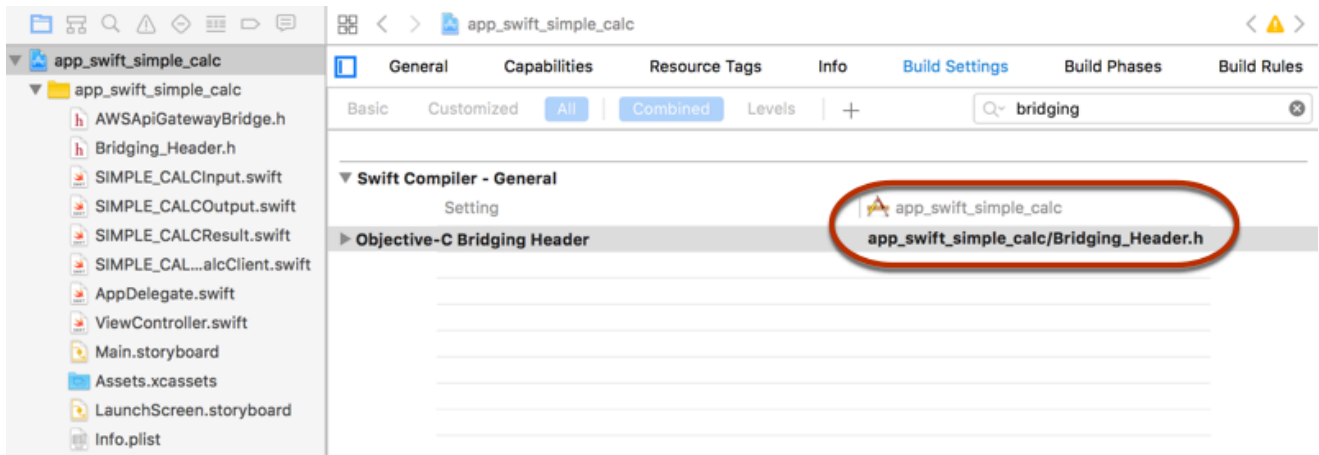
```
pod install
```

Isso instala o componente do API Gateway e quaisquer componentes dependentes do SDK móvel da AWS no projeto da aplicação.

- d. Feche o projeto Xcode e abra o arquivo \*.xcworkspace para reiniciar o Xcode.
- e. Adicione todos os arquivos de cabeçalho do SDK (.h) e arquivos de código-fonte Swift (.swift) do diretório extraído generated-src para seu projeto Xcode.



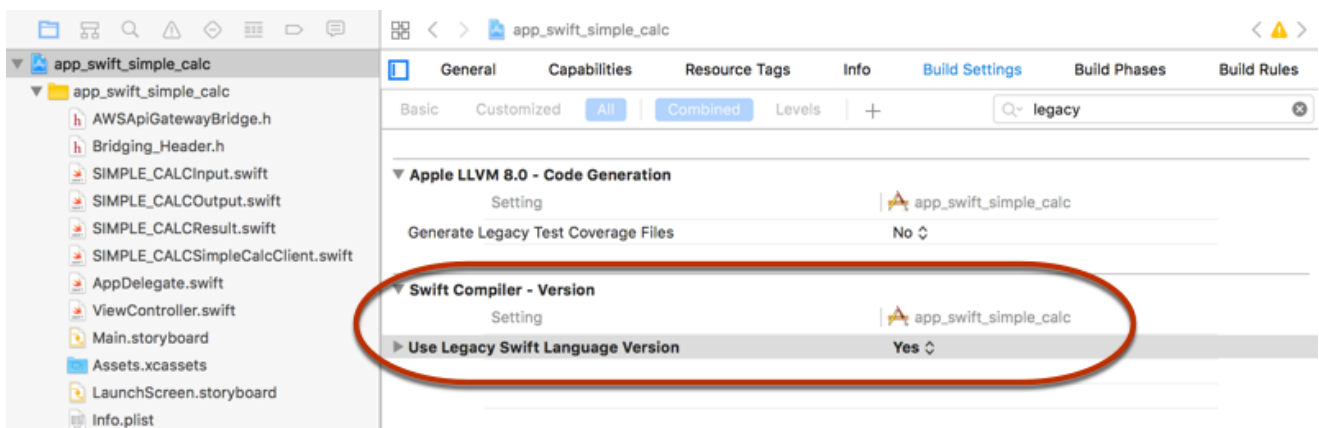
- f. Para permitir a chamada de bibliotecas Objective-C do SDK Móvel da AWS a partir do seu projeto de código Swift, defina o caminho do arquivo Bridging\_Header.h na propriedade Objective-C Bridging Header (Cabeçalho ponte Objective-C), na definição Swift Compiler - General (Compilador Swift - Geral) da configuração do projeto Xcode:



**Tip**

Você pode digitar **bridging** na caixa de pesquisa do Xcode para localizar a propriedade Objective-C Bridging Header (Cabeçalho ponte Objective-C).

- g. Construa o projeto Xcode para verificar se ele está corretamente configurado antes de prosseguir. Se o seu Xcode usar uma versão mais recente do Swift do que a versão com suporte para o SDK Móvel da AWS, você receberá erros do compilador Swift. Nesse caso, defina a propriedade Use Legacy Swift Language Version (Usar versão de linguagem do Swift legado) para Yes (Sim), na configuração Swift Compiler - Version (Compilador Swift - Versão):



Para importar o AWS Mobile SDK for iOS em Swift no seu projeto fazendo download explicitamente do AWS Mobile SDK ou usando o [Carthage](#), siga as instruções no arquivo README.md que acompanha o pacote do SDK. Certifique-se de usar apenas uma dessas opções para importar o SDK Móvel da AWS.

Chamar métodos de API por meio do SDK do iOS gerado pelo API Gateway em um projeto Swift

Quando você gerou o SDK com o prefixo de `SIMPLE_CALC` para essa API [SimpleCalc](#) com dois modelos para descrever a entrada (`Input`) e a saída (`Result`) das solicitações e respostas da API, no SDK, a classe de cliente de API resultante torna-se `SIMPLE_CALCSimpleCalcClient` e as classes de dados correspondentes são `SIMPLE_CALCInput` e `SIMPLE_CALCResult`, respectivamente. As solicitações e respostas da API são mapeadas para os métodos do SDK, da seguinte maneira:

- A solicitação de API de

```
GET /?a=...&b=...&op=...
```

torna-se o método SDK de

```
public func rootGet(op: String?, a: String?, b: String?) -> AWSTask
```

A propriedade `AWSTask.result` é do tipo `SIMPLE_CALCResult`, se o modelo `Result` foi adicionado à resposta do método. Caso contrário, ela será do tipo `NSDictionary`.

- Essa solicitação de API de

```
POST /

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

torna-se o método SDK de

```
public func rootPost(body: SIMPLE_CALCInput) -> AWSTask
```

- A solicitação de API de

```
GET /{a}/{b}/{op}
```

torna-se o método SDK de

```
public func aB0pGet(a: String, b: String, op: String) -> AWSTask
```

O procedimento a seguir descreve como chamar os métodos de API no código-fonte do aplicativo Swift; por exemplo, como parte do `viewDidLoad()` delegado em um arquivo `ViewController.m`.

Como chamar a API por meio do SDK do iOS gerado pelo API Gateway

1. Instancie a classe de cliente da API:

```
let client = SIMPLE_CALCSimpleCalcClient.default()
```

Para usar o Amazon Cognito com a API, defina uma configuração de serviço da AWS padrão (mostrada a seguir) antes de obter o 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. Chame o método GET `/?a=1&b=2&op=+` para realizar 1+2:

```
client.rootGet("+", a: "1", b:"2").continueWithBlock {(task: AWSTask) -> AnyObject?
in
 self.showResult(task)
 return nil
}
```

em que a função auxiliar `self.showResult(task)` imprime o resultado ou o erro no console; por exemplo:

```
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)")
 }
}
}

```

Em um aplicativo de produção, você pode exibir o resultado ou erro em um campo de texto. A exibição resultante é  $1 + 2 = 3$ .

3. Chame a carga POST / para realizar 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
}

```

A exibição resultante é  $1 - 2 = -1$ .

4. Chame GET /{a}/{b}/{op} para realizar 1/2:

```

client.aBOpGet("1", b:"2", op:"div").continueWithBlock {(task: AWSTask) ->
AnyObject? in
 self.showResult(task)
 return nil
}

```

A exibição resultante é  $1 \text{ div } 2 = 0.5$ . Aqui, div é usado no lugar de / porque a [função do Lambda simples](#) no backend não manuseia variáveis de caminho codificadas por URL.

## Desenvolver APIs REST usando OpenAPI no API Gateway

Você pode usar o API Gateway para importar uma API REST de um arquivo de definição externo para o API Gateway. No momento, o API Gateway é compatível com arquivos de definição [v2.0 da OpenAPI](#) e [v3.0 da OpenAPI](#), com exceções listadas em [Notas importantes do Amazon API](#)



[Gateway para APIs REST](#). É possível atualizar uma API substituindo-a por uma nova definição ou mesclar uma definição com uma API existente. As opções são especificadas usando um parâmetro de consulta mode no URL solicitado.

Para obter um tutorial sobre o uso do recurso de importação de API no console do API Gateway, consulte [Tutorial: Criar uma API REST importando um exemplo](#).

#### Tópicos

- [Importar uma API otimizada para bordas para o API Gateway](#)
- [Importar uma API regional para o API Gateway](#)
- [Importar um arquivo do OpenAPI para atualizar uma definição de API existente](#)
- [Defina a propriedade basePath OpenAPI](#)
- [AWS de variáveis para importação de OpenAPI](#)
- [Erros e avisos ao importar a API para o API Gateway](#)
- [Exportar uma API REST do API Gateway](#)

### Importar uma API otimizada para bordas para o API Gateway

Você pode importar um arquivo de definição do OpenAPI de uma API para criar uma nova API otimizada para fronteiras especificando o tipo de endpoint EDGE como uma entrada adicional, além do arquivo do OpenAPI, para a operação de importação. É possível fazer isso usando o console do API Gateway, a AWS CLI ou um AWS SDK.

Para obter um tutorial sobre o uso do recurso Import API no console do API Gateway, consulte [Tutorial: Criar uma API REST importando um exemplo](#).

#### Tópicos

- [Importar uma API otimizada para bordas usando o console do API Gateway](#)
- [Importar uma API otimizada para bordas usando a AWS CLI](#)

### Importar uma API otimizada para bordas usando o console do API Gateway

Para importar uma API otimizada para bordas usando o console do API Gateway, faça o seguinte:

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione Create API (Criar API).

3. Em REST API (API REST), escolha Import (Importar).
4. Copie uma definição do OpenAPI da API e cole-a no editor de código ou selecione Escolher arquivo para carregar um arquivo do OpenAPI de uma unidade local.
5. Em Tipo de endpoint da API, escolha Otimizado para borda.
6. Escolha Criar API para começar a importar as definições do OpenAPI.

### Importar uma API otimizada para bordas usando a AWS CLI

O comando [import-rest-api](#) indicado abaixo importa uma API do arquivo de definição do OpenAPI para criar uma API otimizada para borda:

```
aws apigateway import-rest-api \
 --fail-on-warnings \
 --body 'file://path/to/API_OpenAPI_template.json'
```

ou com uma especificação explícita do parâmetro de string de consulta `endpointConfigurationTypes` para EDGE:

```
aws apigateway import-rest-api \
 --parameters endpointConfigurationTypes=EDGE \
 --fail-on-warnings \
 --body 'file://path/to/API_OpenAPI_template.json'
```

### Importar uma API regional para o API Gateway

Ao importar uma API, você pode escolher a configuração de endpoint regional para a API. É possível usar o console do API Gateway, a AWS CLI ou um AWS SDK.

Ao exportar uma API, a configuração de endpoint da API não está incluído nas definições da API exportada.

Para obter um tutorial sobre o uso do recurso Import API no console do API Gateway, consulte [Tutorial: Criar uma API REST importando um exemplo](#).

### Tópicos

- [Importar uma API regional usando o console do API Gateway](#)
- [Importar uma API regional usando a AWS CLI](#)

## Importar uma API regional usando o console do API Gateway

Para importar uma API de um endpoint regional usando o console do API Gateway, faça o seguinte:

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione Create API (Criar API).
3. Em REST API (API REST), escolha Import (Importar).
4. Copie uma definição do OpenAPI da API e cole-a no editor de código ou selecione Escolher arquivo para carregar um arquivo do OpenAPI de uma unidade local.
5. Em Tipo de endpoint da API, escolha Regional.
6. Escolha Criar API para começar a importar as definições do OpenAPI.

## Importar uma API regional usando a AWS CLI

O comando [import-rest-api](#) indicado abaixo importa um arquivo de definição OpenAPI e define o tipo de endpoint como regional:

```
aws apigateway import-rest-api \
 --parameters endpointConfigurationTypes=REGIONAL \
 --fail-on-warnings \
 --body 'file:///path/to/API_OpenAPI_template.json'
```

## Importar um arquivo do OpenAPI para atualizar uma definição de API existente

Você pode importar as definições de APIs apenas para atualizar uma API existente, sem alterar sua configuração de endpoint, bem como os estágios e suas variáveis ou referências a chaves de APIs.

A operação de importação para atualização pode ocorrer em dois modos: mesclagem ou substituição.

Quando uma API (A) é mesclada com outra (B), a API resultante retém as definições de A e B, se as duas APIs não compartilharem definições conflitantes. Em caso de conflito, as definições de método da API de mesclagem (A) substituem as definições de método correspondentes da API mesclada (B). Por exemplo, suponha que B tenha declarado os seguintes métodos para retornar as respostas 200 e 206:

```
GET /a
POST /a
```

e que A declare o seguinte método para retornar as respostas 200 e 400:

```
GET /a
```

Quando A é mesclada com B, a API resultante produz os seguintes métodos:

```
GET /a
```

que retorna 200 e 400 respostas e

```
POST /a
```

que retorna 200 e 206 respostas.

A mesclagem de uma API é útil quando você decompõe suas definições de API externas em várias partes menores e apenas deseja aplicar as alterações de uma dessas partes de cada vez. Por exemplo, isso pode ocorrer se várias equipes são responsáveis por diferentes partes de uma API e têm alterações disponíveis em ritmos diferentes. Nesse modo, os itens da API existente que não estiverem especificamente indicados na definição importada serão ignorados.

Quando uma API (A) substitui outra API (B), a API resultante assume as definições da API de substituição (A). A substituição de uma API é útil quando uma definição de API externa contém a definição completa de uma API. Nesse modo, os itens de uma API existente que não estiverem especificamente indicados na definição importada serão excluídos.

Para mesclar uma API, envie uma solicitação PUT para `https://apigateway.<region>.amazonaws.com/restapis/<restapi_id>?mode=merge`. O valor do parâmetro do caminho `restapi_id` especifica a API com a qual a definição de API fornecida será mesclada.

O seguinte trecho de código mostra um exemplo da solicitação PUT para mesclar uma definição de API do OpenAPI em JSON, como a carga útil, com a API especificada já no 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](#)

A operação de atualização se mesclagem usa duas definições de API completas e as mescla em uma só. Para uma alteração pequena e incremental, você pode usar a operação de [atualização de recursos](#).

Para substituir uma API, envie uma solicitação PUT para `https://apigateway.<region>.amazonaws.com/restapis/<restapi_id>?mode=overwrite`. O parâmetro de caminho `restapi_id` especifica a API que será substituída pelas definições de API fornecidas.

O seguinte trecho de código mostra um exemplo de uma solicitação de substituição com a carga de uma definição do OpenAPI formatada em JSON:

```
PUT /restapis/<restapi_id>?mode=overwrite
Host:apigateway.<region>.amazonaws.com
Content-Type: application/json
Content-Length: ...
```

### [An OpenAPI API definition in JSON](#)

Quando o parâmetro de consulta `mode` não é especificado, supõe-se uma mesclagem.

#### Note

As operações PUT são idempotentes, mas não atômicas. Isso significa que, se um erro de sistema ocorrer no meio do processamento, a API poderá terminar em mau estado. No entanto, repetir a operação com êxito coloca a API no mesmo estado final, como se a primeira operação tivesse sido bem-sucedida.

## Defina a propriedade **basePath** OpenAPI

No [OpenAPI 2.0](#), você pode usar a propriedade `basePath` para fornecer uma ou mais partes de caminhos que precedem cada caminho definido na propriedade `paths`. Como o API Gateway tem

várias maneiras de expressar um caminho de um recurso, o recurso Import API (Importar API) fornece as seguintes opções para interpretar a propriedade `basePath` durante uma importação: ignorar, preceder e dividir.

No [OpenAPI 3.0](#), `basePath` não é mais uma propriedade de nível superior. Em vez disso, o API Gateway usa uma [variável de servidor](#) como convenção. O recurso Import API (Importar API) fornece as mesmas opções para interpretar o caminho base durante a importação. O caminho base é identificado da seguinte forma:

- Se a API não contém variáveis `basePath`, o recurso Import API (Importar API) verifica a string `server.url` para conferir se ela contém um caminho além de `"/`. Se sim, esse caminho será usado como caminho base.
- Se a API contém apenas uma variável `basePath`, o recurso Import API (Importar API) a usa como caminho base, mesmo que ela não seja indicada no `server.url`.
- Se a API contém várias variáveis `basePath`, o recurso Import API (Importar API) usa apenas a primeira como caminho base.

### Ignorar

Se o arquivo do OpenAPI tiver um valor `basePath` de `/a/b/c`, e a propriedade `paths` contiver `/e` e `/f`, a seguinte solicitação POST ou PUT:

```
POST /restapis?mode=import&basepath=ignore
```

```
PUT /restapis/api_id?basepath=ignore
```

resultará nos seguintes recursos na API:

- `/`
- `/e`
- `/f`

O efeito é tratar `basePath` como se ele não estivesse presente, e todos os recursos da API declarados são atendidos em relação ao host. Isso pode ser usado, por exemplo, quando você tem um nome de domínio personalizado com um mapeamento de API que não inclui um caminho base e um valor de estágio que faz referência ao seu estágio de produção.

 Note

O API Gateway cria automaticamente um recurso raiz para você, mesmo que ele não esteja explicitamente declarado no seu arquivo de definição.

Quando não especificado, o `basePath` pega ignore por padrão.

### Preceder

Se o arquivo do OpenAPI tiver um valor `basePath` de `/a/b/c` e a propriedade `paths` contiver `/e` e `/f`, a seguinte solicitação POST ou PUT:

```
POST /restapis?mode=import&basepath=prepend
```

```
PUT /restapis/api_id?basepath=prepend
```

resultará nos seguintes recursos na API:

- `/`
- `/a`
- `/a/b`
- `/a/b/c`
- `/a/b/c/e`
- `/a/b/c/f`

O efeito é tratar `basePath` como se recursos adicionais estivessem sendo especificados (sem métodos) e adicioná-los ao conjunto de recursos declarados. Isso pode ser usado, por exemplo, quando diferentes equipes são responsáveis por diferentes partes de uma API e o `basePath` pode fazer referência ao local do caminho para a parte da API de cada equipe.

 Note

O API Gateway cria recursos intermediários automaticamente para você, mesmo que eles não estejam explicitamente declarados na sua definição.

## Split

Se o arquivo do OpenAPI tiver um valor `basePath` de `/a/b/c` e a propriedade `paths` contiver `/e` e `/f`, a seguinte solicitação POST ou PUT:

```
POST /restapis?mode=import&basepath=split
```

```
PUT /restapis/api_id?basepath=split
```

resultará nos seguintes recursos na API:

- `/`
- `/b`
- `/b/c`
- `/b/c/e`
- `/b/c/f`

O efeito é tratar a parte do caminho mais no início, `/a`, como o início do caminho de cada recurso e criar recursos adicionais (sem método) na própria API. Isso pode ser usado, por exemplo, quando `a` é um nome de estágio que deseja expor como parte da sua API.

## AWS de variáveis para importação de OpenAPI

Você pode usar as seguintes variáveis da AWS nas definições do OpenAPI. O API Gateway resolve as variáveis quando a API é importada. Para especificar uma variável, use `${variable-name}`. A tabela a seguir descreve as estatísticas da AWS disponíveis.

| Nome da variável            | Descrição                                                       |  |
|-----------------------------|-----------------------------------------------------------------|--|
| <code>AWS::AccountId</code> | O ID da conta AWS que importa a API. Por exemplo, 123456789012. |  |
| <code>AWS::Partition</code> | A partição da AWS na qual a API é importada. Para               |  |



| Nome da variável | Descrição                                                           |  |
|------------------|---------------------------------------------------------------------|--|
|                  | Regiões padrão da AWS a partição é aws.                             |  |
| AWS::Region      | A região da AWS na qual a API é importada. Por exemplo, us-east-2 . |  |

## AWSExemplo de variáveis da

O exemplo a seguir usa variáveis da AWS para especificar uma função do AWS Lambda para uma integração.

### 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
```

## Erros e avisos ao importar a API para o API Gateway

Ao importar o arquivo de definição externo para o API Gateway, ele pode gerar avisos e erros. As seções a seguir explicam os erros e os avisos que podem ocorrer durante a importação.

### Erros durante a importação

Durante a importação, erros podem ser gerados para problemas importantes, como um documento inválido do OpenAPI. Os erros são retornados como exceções (por exemplo, `BadRequestException`) em uma resposta mal-sucedida. Quando ocorre um erro, a nova definição da API é descartada, e nenhuma alteração é feita na API existente.

### Avisos durante a importação

Durante a importação, avisos podem ser gerados para problemas menores, como uma referência de modelo ausente. Se um aviso ocorrer, a operação continuará se a expressão de consulta `failonwarnings=false` for acrescentada à URL da solicitação. Caso contrário, as atualizações serão revertidas. Por padrão, `failonwarnings` é definido como `false`. Nesses casos, os avisos são retornados como um campo no recurso [RestApi](#) resultante. Caso contrário, os avisos serão retornados como uma mensagem na exceção.

## Exportar uma API REST do API Gateway

Depois de criar e configurar uma API REST no API Gateway, usando o console do API Gateway ou de outra forma, você pode exportá-la para um arquivo OpenAPI usando a API de exportação do API Gateway, que faz parte do Amazon API Gateway Control Service. Para usar a API de exportação do API Gateway, você precisa assinar suas solicitações de API. Para ter mais informações sobre como assinar solicitações, consulte [Assinar solicitações de API](#) no Guia do usuário do IAM. Existem opções para incluir as extensões de integração do API Gateway, bem como as extensões do [Postman](#), no arquivo de definição do OpenAPI exportado.

### Note

Ao exportar a API usando a AWS CLI, inclua o parâmetro de extensões, conforme mostrado no exemplo a seguir, para garantir que a extensão `x-amazon-apigateway-request-validator` seja incluída:

```
aws apigateway get-export --parameters extensions='apigateway' --rest-api-id
 abcdefg123 --stage-name dev --export-type swagger latestswagger2.json
```

Não será possível exportar uma API se suas cargas não forem do tipo `application/json`. Se você tentar, receberá uma resposta de erro indicando que os modelos de corpo JSON não foram encontrados.

### Solicitar a exportação de uma API REST

Com o recurso Export API, você exporta uma API REST existente ao enviar uma solicitação GET, especificando a API a ser exportada como parte de caminhos de URL. A URL da solicitação tem o seguinte 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
```

Você pode acrescentar a string de consulta `extensions` para especificar se deseja incluir extensões do API Gateway (com o valor `integration`) ou extensões do Postman (com o valor `postman`).

Além disso, é possível definir o cabeçalho `Accept` como `application/json` ou `application/yaml` para receber a saída da definição de API no formato JSON ou YAML, respectivamente.

Para obter mais informações sobre como enviar solicitações GET usando a API Export do API Gateway, consulte [GetExport](#).

#### Note

Se você definir modelos na sua API, eles deverão ser destinados ao tipo de conteúdo “`application/json`” para que o API Gateway exporte o modelo. Caso contrário, o API Gateway lançará uma exceção com a mensagem de erro de que apenas modelos de corpo não JSON foram encontrados.

Os modelos devem conter propriedades ou serem definidos como um determinado tipo de `JSONSchema`.

Fazer download da definição da API REST do OpenAPI em JSON

Para exportar e fazer download de uma API REST em definições do OpenAPI no 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
```

Aqui, **<region>** poderia ser, por exemplo, `us-east-1`. Para conhecer todas as regiões onde o API Gateway está disponível, consulte [Regiões e endpoints](#).

Fazer download da definição da API REST do OpenAPI em YAML

Para exportar e fazer download de uma API REST em definições do OpenAPI no 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
```

Fazer download da definição da API REST do OpenAPI com extensões Postman em JSON

Para exportar e fazer download de uma API REST em definições do OpenAPI com Postman no 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
```

Fazer download da definição da API REST do OpenAPI com integração ao API Gateway em YAML

Para exportar e fazer download de uma API REST em definições do OpenAPI com integração ao API Gateway no formato YAML:

## OpenAPI 3.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/oas30?extensions=integrations
Host: apigateway.<region>.amazonaws.com
Accept: application/yaml
```

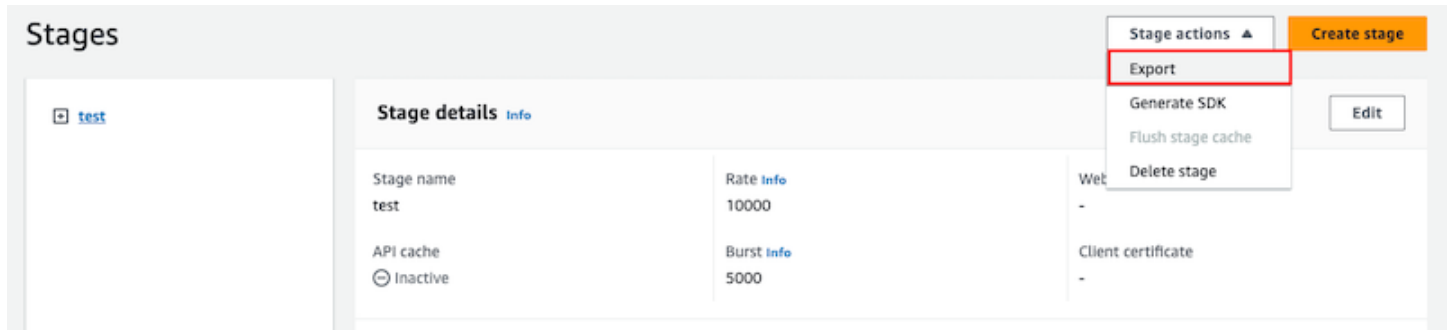
## OpenAPI 2.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/swagger?
extensions=integrations
Host: apigateway.<region>.amazonaws.com
Accept: application/yaml
```

## Exportar API REST usando o console do API Gateway

Depois de [implantar sua API REST em um estágio](#), você pode prosseguir e exportar essa API no estágio para um arquivo do OpenAPI usando o console do API Gateway.

No painel Estágios no console do API Gateway, escolha Estágios, Exportar.



Especifique Tipo de especificação da API, Formato e Extensões para baixar a definição do OpenAPI da sua API.

## Publicar APIs REST para os clientes invocarem

Apenas criar e desenvolver uma API do API Gateway não fará com que ela possa ser chamada automaticamente pelos usuários. Para que ela possa ser chamada, implante a API em um estágio. Além disso, você pode querer personalizar o URL que os usuários usarão para acessar a API. Você pode dar a ele um domínio que seja consistente com sua marca ou que seja mais memorável que o URL padrão da API.

Nesta seção, você pode aprender a implantar a API e personalizar o URL fornecido aos usuários para acessá-la.

### Note

Para aumentar a segurança das APIs do API Gateway, o domínio `execute-api.{region}.amazonaws.com` é registrado na [Public Suffix List \(PSL\)](#). Para maior segurança, recomendamos que você use cookies com um prefixo `__Host-` se precisar definir cookies confidenciais no nome de domínio padrão para as APIs do API Gateway. Essa prática ajudará a defender seu domínio contra tentativas de falsificação de solicitação entre sites (CSRF). Para obter mais informações, consulte a página [Set-Cookie](#) na Mozilla Developer Network.

## Tópicos

- [Implantar APIs REST no API Gateway](#)
- [Nome de domínio personalizado para APIs REST públicas no API Gateway](#)

## Implantar APIs REST no API Gateway

Depois de criar sua API, você deve implantá-la para permitir que seja chamada por seus usuários.

Para implantar uma API, você cria uma implantação da API e a associa a um estágio. Um estágio é uma referência lógica a um estado do ciclo de vida de sua API (por exemplo, dev, prod, beta, v2). Os estágios de API são identificados pelo ID da API e pelo nome do estágio. Eles são incluídos no URL que você usa para chamar a API. Cada estágio é uma referência nomeada a uma implantação da API e é disponibilizado para chamadas feitas por aplicativos cliente.

### Important

Toda vez que atualizar uma API, você deve reimplantar a API em um estágio existente ou em um novo estágio. A atualização de uma API inclui a modificação de rotas, métodos, integrações, autorizadores, políticas de recursos e qualquer outro elemento além das configurações do estágio.

À medida que a sua API evolui, você pode continuar a implantá-la em diferentes estágios como versões distintas da API. Você também pode implantar suas atualizações de API como uma [implantação de lançamento canary](#). Isso permite que seus clientes de API acessem, no mesmo estágio, a versão de produção por meio da versão de produção e a versão atualizada por meio da versão canary.

Para chamar uma API implantada, o cliente envia uma solicitação com base na URL da API. A URL é determinada pelo protocolo de uma API (HTTP(S) ou (WSS)), nome do host, nome do estágio e (para APIs REST) caminho de recurso. O nome do host e o nome do estágio definem o URL base da API.

Com o nome de domínio padrão da API, o URL base de uma API REST (por exemplo) em determinado estágio (*{stageName}*) tem o seguinte formato:

```
https://{restapi-id}.execute-api.{region}.amazonaws.com/{stageName}
```

Para facilitar o uso do URL base padrão da API, você pode criar um nome de domínio personalizado (por exemplo, api.example.com) para substituir o nome de domínio padrão da API. Para oferecer suporte a várias APIs sob o nome de domínio personalizado, você deve mapear um estágio de API para um caminho de base.



Com um nome de domínio personalizado `{api.example.com}` e o estágio de API mapeado para um caminho base (`{basePath}`) sob o nome de domínio personalizado, a URL base de uma API REST se torna a seguinte:

```
https://{api.example.com}/{basePath}
```

Para cada estágio, você pode otimizar o a performance da API ajustando os limites de controle de fluxo de solicitações em nível de conta padrão e habilitando o armazenamento em cache da API. Você também pode habilitar o registro em log de chamadas de API no CloudTrail ou no CloudWatch e selecionar um certificado de cliente para o backend autenticar as solicitações da API. Além disso, você pode substituir as configurações em nível de estágio para os métodos individuais e definir as variáveis de estágio para passar contextos de ambiente específicos de estágio para a integração da API em tempo de execução.

Os estágios permitem um controle de versão robusto para sua API. Por exemplo, você pode implantar uma API em um estágio test e um prod, e usar o estágio test como uma compilação de teste e o estágio prod como uma compilação estável. Depois que as atualizações passarem no teste, você poderá promover o estágio test para o estágio prod. A promoção pode ser feita por meio da reimplantação da API no estágio prod ou da atualização do valor de uma variável de estágio do nome de estágio test para prod.

Nesta seção, discutiremos como implantar uma API usando o [console do API Gateway](#) ou chamando a [API REST do API Gateway](#). Para usar outras ferramentas, consulte a documentação da [CLI da AWS](#) ou um [SDK da AWS](#).

## Tópicos

- [Criar uma implantação para uma API REST no API Gateway](#)
- [Configurar um estágio para uma API REST no API Gateway](#)
- [Configurar uma implantação de versão canary do API Gateway](#)
- [Atualizações das APIs REST que exigem reimplantação](#)

## Criar uma implantação para uma API REST no API Gateway

No API Gateway, uma implantação de API REST é representada por um recurso [Deployment](#). Ele é semelhante a um executável de uma API que é representada por um recurso [RestApi](#).

Para o cliente chamar a API, você deve criar uma implantação e associar um estágio a ela. Um estágio é representado por um recurso [Stage](#). Ele representa um snapshot da API, incluindo

métodos, integrações, modelos, modelos de mapeamento e autorizadores do Lambda (anteriormente conhecidos como autorizadores personalizados). Quando você atualiza a API, pode reimplantá-la associando um novo estágio à implantação existente. Discutimos a criação de um estágio em [the section called “Configurar um estágio”](#).

## Tópicos

- [Crie um implantação do .](#)
- [Próximas etapas para sua implantação de API](#)

## Crie um implantação do .

Os procedimentos a seguir mostram como criar uma implantação para uma API REST.

## Console de gerenciamento da AWS

Você deve ter criado uma API REST antes de implantá-la pela primeira vez. Para ter mais informações, consulte [Desenvolver APIs REST no API Gateway](#).

O console do API Gateway permite que você implante uma API criando uma implantação e associando-a a um estágio novo ou existente.

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação APIs, escolha a API que você deseja implantar.
3. No painel Resources (Recursos), escolha Deploy API (Implantar API).
4. Em Estágio, selecione uma das seguintes opções:
  - a. Para criar um estágio, selecione Novo estágio e insira um nome em Nome do estágio. Opcionalmente, você pode fornecer uma descrição para a implantação em Descrição da implantação.
  - b. Para escolher um estágio existente, selecione o nome dele no menu suspenso. É aconselhável fornecer uma descrição para a nova implantação em Descrição da implantação.
  - c. Para criar uma implantação que não esteja associada a um estágio, selecione Sem estágio. Posteriormente, você poderá associar essa implantação a um estágio.
5. Escolha Implantar.

## AWS CLI

Ao criar uma implantação, você instancia o recurso [Deployment](#).

O comando [create-deployment](#) indicado abaixo cria uma implantação:

```
aws apigateway create-deployment --rest-api-id rest-api-id
```

A API não poderá ser chamada enquanto essa implantação não for associada a um estágio. Com um estágio existente, você pode fazer isso atualizando a propriedade [deploymentId](#) do estágio com o ID de implantação recém-criado. O comando [update-stage](#) indicado abaixo atualiza o estágio com uma nova implantação: No console, isso é chamado de Implantação ativa.

```
aws apigateway update-stage \
 --rest-api-id rest-api-id \
 --stage-name 'stage-name' \
 --patch-operations op='replace',path='/deploymentId',value='deployment-id'
```

Ao criar uma implantação, você também pode associá-la simultaneamente a um novo estágio. O comando [create-deployment](#) indicado abaixo cria uma implantação e a associa a um novo estágio chamado beta:

```
aws apigateway create-deployment \
 --rest-api-id rest-api-id \
 --stage-name beta
```

Para reimplantar uma API, execute as mesmas etapas. Você pode reutilizar o mesmo estágio.

### Próximas etapas para sua implantação de API

Veja a seguir as próximas etapas para a implantação da sua API.

### Modificar as configurações de armazenamento

Depois que uma API é implantada, é possível modificar as configurações de estágio para habilitar ou desabilitar o cache, o registro em log ou a limitação de solicitações dessa API. Você também pode escolher um certificado de cliente para o backend autenticar o API Gateway e definir variáveis de estágio para transmitir o contexto de implantação para a integração da API em tempo de execução. Para ter mais informações, consulte [Modifique as configurações de armazenamento](#).

Depois de modificar as configurações do estágio, você deve reimplantar a API para que as alterações entrem em vigor.

 **Note**

Se as configurações atualizadas, como a habilitação de registro em log, exigirem uma nova função do IAM, você poderá adicionar a função do IAM necessária sem reimplantar a API. No entanto, pode demorar alguns minutos para o novo perfil do IAM entrar em vigor. Antes que isso aconteça, os rastreamentos das suas chamadas de API não são registrados em log, mesmo que você tenha habilitado a opção de registro em log.

## Escolher diferentes combinações de estágios de implantação

Como uma implantação representa um snapshot de API e um estágio define um caminho em um snapshot, você pode escolher diferentes combinações de estágio de implantação para controlar como os usuários invocam diferentes versões da API. Isso é útil, por exemplo, quando você deseja reverter o estado da API para uma implantação anterior ou mesclar uma "ramificação particular" da API na ramificação pública.

O procedimento a seguir mostra como fazer isso usando o Stage Editor (Editor de estágio) no console do API Gateway. Supõe-se que você tenha implantado uma API mais de uma vez.

1. Se você ainda não estiver no painel Estágios, no painel de navegação principal, escolha Estágios.
2. Selecione o estágio que você deseja atualizar.
3. Na guia Histórico de implantação, escolha a implantação que o estágio deve usar.
4. Escolha Alterar implantação ativa.
5. Confirme que você deseja alterar a implantação ativa e escolha Alterar implantação ativa na caixa de diálogo Tornar implantação ativa.

Passe dados específicos da implantação para sua API.

Para uma implantação, é possível definir ou modificar variáveis de estágio para transmitir dados específicos da implantação à integração da API em tempo de execução. Você pode fazer isso na guia Variáveis de estágio no Editor de estágio. Para obter mais informações, consulte as instruções em [Usar variáveis de estágio para uma API REST no API Gateway](#).

## Configurar um estágio para uma API REST no API Gateway

Um estágio é uma referência designada para uma implantação, que é um snapshot da API. Use um [Stage](#) para gerenciar e otimizar uma implantação específica. Por exemplo, você pode definir configurações do estágio para ativar o cache, personalizar a limitação de solicitação, configurar o registro em log, definir variáveis do estágio ou anexar uma versão de canary para testes. A seção a seguir mostra como criar e configurar seu palco.

### Criar um novo estágio

Após a implantação inicial, você pode adicionar mais estágio e associá-los a implantações existentes. Você pode usar o console do API Gateway para criar um estágio ou escolher um estágio existente ao implantar uma API. Em geral, você pode adicionar um novo estágio a uma implantação de API antes de reimplantar essa API. Para criar um estágio usando o console do API Gateway, siga estas etapas:

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST.
3. No painel de navegação principal, selecione Estágios em uma API.
4. No painel de navegação Estágios, selecione Criar estágio.
5. Em Nome do estágio, insira um nome; por exemplo, **prod**.

#### Note

Nomes de estágio podem conter apenas caracteres alfanuméricos, hífen e sublinhados. O tamanho máximo é de 128 caracteres.

6. (Opcional). Em Descrição, insira uma descrição do estágio.
7. Em Implantação, selecione a data e a hora da implantação de API existente que você deseja associar a esse estágio.
8. Em Configurações adicionais, é possível especificar configurações adicionais para o estágio.
9. Selecione Criar estágio.

Modifique as configurações de armazenamento.

Depois de uma implantação bem-sucedida de uma API, o estágio é preenchido com as configurações padrão. É possível usar o console ou a API REST do API Gateway para alterar as

configurações de estágio, incluindo o registro em log e o armazenamento em cache de APIs. Se você modificou o endpoint padrão da API REST, a modificação será propagada para todos os estágios da API ao atualizar um estágio. As etapas a seguir mostram como fazer isso usando o Editor de estágio do console do API Gateway.

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST.
3. No painel de navegação principal, selecione Estágios em uma API.
4. No painel Estágios, escolha o nome do estágio.
5. Na seção Detalhes do estágio, selecione Editar.
6. (Opcional) Em Descrição do estágio, edite a descrição.
7. Em Configurações adicionais, você vai modificar as seguintes configurações:

### Configurações de cache

Para habilitar o armazenamento em cache da API para o estágio, ative Provisionar cache de APIs. Depois, configure Armazenamento em cache padrão no nível de método, Capacidade do cache, Criptografar dados do cache, Tempo de vida (TTL) do cache, bem como os requisitos para invalidação de cache por chave.

O armazenamento em cache não estará ativo até você ativar o cache padrão no nível de método ou ativar o cache no nível de método para um método específico.

Para obter mais informações sobre as configurações de cache, consulte [Configurações de cache para APIs REST no API Gateway](#).

#### Note

Se você habilitar o armazenamento em cache da API para um estágio da API, sua conta da AWS poderá ser cobrada por armazenamento em cache da API. O armazenamento em cache não está qualificado para o nível gratuito da AWS.

### Configurações de controle de utilização

Para definir destinos de controle de utilização em nível de estágio para todos os métodos associados a essa API, ative Controle de utilização.

Em Rate (Taxa), insira uma taxa alvo. Essa é a taxa, em solicitações por segundo, em que os tokens são adicionados ao bucket do token. Essa taxa em nível de estágio não deve ser superior à taxa em [nível de conta](#), conforme especificado em [Cotas para configurar e executar uma API REST no API Gateway](#).

Em Burst (Intermitência), insira uma taxa de intermitência alvo. A taxa de expansão é a capacidade do bucket do token. Isso permite mais solicitações por um período do que a taxa alvo. Essa taxa de intermitência no nível do estágio não deve ser superior à taxa de intermitência no [nível da conta](#), conforme especificado em [Cotas para configurar e executar uma API REST no API Gateway](#).

 Note

As taxas de controle de utilização não são limites rígidos e são aplicadas de acordo com os melhores esforços. Em alguns casos, os clientes podem exceder os alvos definidos por você. Não dependa do controle de utilização para controlar custos ou bloquear o acesso a uma API. Considere usar [AWS Budgets](#) para monitorar custos e [AWS WAF](#) para gerenciar solicitações de APIs.

## Configurações de firewall e certificado

Para associar uma ACL da web do AWS WAF ao estágio, selecione uma ACL da web na lista suspensa Web ACL. Se desejar, selecione Bloquear solicitações de API se não for possível avaliar a WebACL (Falha - Fechar).

Para selecionar um certificado de cliente para o estágio, selecione um certificado no menu suspenso Certificado do cliente.

8. Escolha Continuar.
9. Revise as alterações e selecione Salvar alterações.
10. Para habilitar o Amazon CloudWatch Logs para todos os métodos associados a esse estágio da API do API Gateway, na seção Logs e rastreamento, selecione Editar.

 Note

Para habilitar o CloudWatch Logs, também é necessário especificar o ARN de um perfil do IAM que permita ao API Gateway gravar informações no CloudWatch Logs em

nome do usuário. Para isso, escolha Settings (Configurações) no painel de navegação principal APIs. Depois, em Perfil do log do CloudWatch, insira o ARN de um perfil do IAM.

Em cenários de aplicações comuns, o perfil do IAM pode anexar a política gerenciada [AmazonAPIGatewayPushToCloudWatchLogs](#).

A função do IAM também deve conter a seguinte instrução de relação de confiança:

JSON

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

Para obter mais informações sobre o CloudWatch, consulte o [Guia do usuário do Amazon CloudWatch](#).

11. Selecione um nível de registro em log no menu suspenso CloudWatch Logs. Os níveis de registro em log são os seguintes:
  - Desativado: o registro em log não está ativado neste estágio.
  - Somente erros: o registro em log está habilitado somente para erros.
  - Logs de erros e informações: o registro em log está habilitado para todos os eventos.
12. Selecione Rastreamento de dados para que o API Gateway apresente ao CloudWatch o registro em log do rastreamento de dados do seu estágio. Isso pode ser útil para solucionar problemas de APIs, mas pode resultar no registro de dados confidenciais.



 Note

Recomendamos não usar Rastreamento de dados para APIs de produção.

13. Selecione Métricas detalhadas para que o API Gateway relate para o CloudWatch as métricas de API de API calls, Latency, Integration latency, 400 errors e 500 errors. Para saber mais sobre o CloudWatch, consulte [Monitoramento básico e monitoramento detalhado](#) no Guia do usuário do Amazon CloudWatch.

 Important

Sua conta será cobrada pelo acesso a métricas do CloudWatch em nível de método, mas não pelas métricas em nível de API ou estágio.

14. Para habilitar o registro em log de acesso a um destino, ative Registro em log de acesso personalizado.
15. Em ARN de destino de logs de acesso, insira o ARN de um grupo de logs ou um fluxo do Firehose.

O formato do ARN para o Firehose é `arn:aws:firehose:{region}:{account-id}:deliverystream/amazon-apigateway-{your-stream-name}`. O nome do fluxo do Firehose deve ser `amazon-apigateway-{your-stream-name}`.

16. Em Formato do log, insira um formato de log. Para saber mais sobre exemplos de formatos de log, consulte [the section called “Formatos de log do CloudWatch para o API Gateway”](#).
17. Para habilitar o rastreamento [AWS X-Ray](#) para o estágio da API, selecione Rastreamento com X-Ray. Para obter mais informações, consulte [Rastrear solicitações de usuário para APIs REST usando o X-Ray no API Gateway](#).
18. Escolha Salvar alterações. Implante a API novamente para que as novas configurações entrem em vigor.

## Substituir configurações em nível de estágio

Depois de personalizar as configurações em nível de estágio, você pode substituí-las para cada método de API. Algumas dessas opções podem gerar cobranças adicionais na Conta da AWS.

1. Para configurar substituições de método, expanda o estágio no painel de navegação secundário e selecione um método.

## Stages

[Stage actions](#) ▼[Create stage](#)

prod

/

GET

/pets

GET

OPTIONS

POST

/{petId}

GET

OPTIONS

### Method overrides

[Edit](#)

By default, methods inherit stage-level settings. To customize settings for a method, configure method overrides.

 This method inherits its settings from the 'prod' stage.

#### Invoke URL

 <https://abcd1234.execute-api.us-west-1.amazonaws.com/prod/pets>

2. Em Substituições de método, selecione Editar.
3. Para ativar as configurações do CloudWatch em nível de método, para o CloudWatch Logs, selecione um nível de registro em log.
4. Para ativar o registro em log do rastreamento de dados referente ao seu método, selecione Rastreamento de dados.

 Note

Recomendamos não usar Rastreamento de dados para APIs de produção.

5. Para ativar as métricas detalhadas em nível de método, selecione Métricas detalhadas. Sua conta será cobrada pelo acesso a métricas do CloudWatch em nível de método, mas não pelas métricas em nível de API ou estágio.
6. Para ativar o controle de utilização em nível de método, selecione Controle de utilização. Insira as opções apropriadas em nível de método. Para saber mais sobre o controle de utilização, consulte [the section called “Limitação”](#).
7. Para configurar o cache no nível de método, selecione Ativar cache de método. Se você alterar a configuração de armazenamento padrão em cache no nível de método nos Detalhes do estágio, isso não afetará essa configuração.
8. Escolha Salvar.

Adicionar uma API REST do API Gateway como destino para o Amazon Bedrock AgentCore Gateway.

O Amazon Bedrock AgentCore Gateway oferece aos desenvolvedores de agentes de IA uma maneira segura de expor as APIs REST do API Gateway como ferramentas compatíveis com o protocolo de contexto para modelos (MCP). O AgentCore Gateway usa destinos para definir ferramentas. Quando você adiciona seu estágio como um destino, o gateway se torna um único URL do MCP que permite acesso às ferramentas de um agente. Para ter mais informações, consulte [Estágios da API REST do API Gateway como destinos](#) no Guia do desenvolvedor do Amazon Bedrock AgentCore Gateway.

Os destinos do API Gateway conectam o AgentCore Gateway aos estágios de suas APIs REST. É possível incluir o estágio inteiro como destino ou selecionar recursos. Depois de criar o destino do API Gateway, o AgentCore Gateway converte as solicitações do MCP recebidas em solicitações HTTP e processa a formatação da resposta. Os clientes MCP podem recuperar a documentação da API usando o método `tools/list` e invocar APIs usando o método `tools/call`.

## Considerações

As considerações a seguir podem impactar seu uso ao adicionar um estágio como destino a um AgentCore Gateway:

- Você já deve ter criado um AgentCore Gateway.
- Somente as APIs REST públicas são compatíveis.
- Não é possível desabilitar o endpoint padrão para a API.
- Cada método da API deve ter um [nome de operação](#) definido para ele ou será necessário criar uma substituição de nome ao adicionar seu estágio como destino. Esse nome é usado como o nome da ferramenta que os agentes usam para interagir com seu método.
- Você pode usar os tipos de provedor de credenciais API\_KEY, NO\_AUTH, ou GATEWAY\_IAM\_ROLE como autenticação de saída para permitir que o gateway acesse a API. O provedor de credenciais API\_KEY é definido pelo AgentCore Gateway. Você pode usar sua chave de API existente do API Gateway. Para ter mais informações, consulte [Configurar autenticação de saída](#).
- Se você usa um grupo de usuários do Amazon Cognito ou um autorizador do Lambda para controlar o acesso à API, os clientes MCP não poderão acessá-la.
- A API deve estar na mesma conta e região do AgentCore Gateway.

### Adicionar um estágio de uma API como destino para um AgentCore Gateway

O procedimento a seguir mostra como adicionar um estágio de uma API como um destino para um AgentCore Gateway.

#### Como adicionar um estágio de uma API como destino para um AgentCore Gateway

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST que esteja implantada em um estágio.
3. No painel de navegação principal, selecione Estágios.
4. Escolha Ações de estágio e Criar destino do MCP.
5. Em AgentCore Gateway, selecione um AgentCore Gateway.
6. Em Nome do destino, insira um nome para o destino.
7. Em Descrição do destino, insira uma descrição.
8. Mantenha a API e o estágio fornecidos.
9. Em Selecione os recursos da API, escolha os recursos da API que os agentes que usam o AgentCore Gateway podem acessar.

Se você não selecionar um recurso, o agente não poderá visualizar a documentação nem invocar o endpoint.

10. A combinação do recurso e do método são as operações da ferramenta. Se a operação não tiver um nome, crie um nome de substituição.

Você também pode definir um nome de operação para um método ao criá-lo.

11. Em Configuração de autenticação de saída, escolha Perfil do IAM, Sem autorização ou Chave de API.
12. Escolha Criar destino.

Para visualizar todos os AgentCore Gateways que têm acesso às suas APIs, escolha a seção Destinos do MCP no painel de navegação principal. Nessa seção, é possível criar um destino do MCP para qualquer API em sua região implantada em um estágio. Escolha Criar destino do MCP e siga as etapas anteriores.

Também é possível visualizar as ferramentas disponíveis para seu destino e editar o destino no console do AgentCore Gateway. Para ter mais informações, consulte [Adicionar destinos a um AgentCore Gateway existente](#).

### Excluir um estágio

Quando você não precisar mais de um estágio, poderá excluí-lo para evitar pagar por recursos não utilizados. As etapas a seguir mostram como usar o console do API Gateway para excluir um estágio.

#### Warning

A exclusão de um estágio pode fazer com que parte ou toda a API correspondente se torne inutilizável pelos autor da chamadas da API. Esse processo não pode ser desfeito, mas você pode recriar o estágio e associá-lo à mesma implantação.

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST.
3. No painel de navegação principal, selecione Estágios.
4. No painel Estágios, escolha o estágio que você deseja excluir e, depois, selecione Ações de estágio, Excluir estágio.
5. Quando solicitado, digite **confirm** e, depois, selecione Excluir.

## Configurar tags para um estágio de API no API Gateway

No API Gateway, você pode adicionar uma tag a um estágio da API, remover a tag do estágio ou visualizá-la. Para fazer isso, use o console do API Gateway, a AWS CLI/SDK ou a API REST do API Gateway.

Um estágio também pode herdar tags de sua API REST pai. Para obter mais informações, consulte [the section called “Herança de tags na API do Amazon API Gateway V1”](#).

Para obter mais informações sobre como marcar recursos do API Gateway, consulte [Atribuição de tags \(tagging\)](#).

### Tópicos

- [Configurar tags para um estágio de API usando o console do API Gateway](#)
- [Configurar tags para um estágio da API usando a AWS CLI](#)
- [Configurar tags para um estágio de API usando a API REST do API Gateway](#)

## Configurar tags para um estágio de API usando o console do API Gateway

O procedimento a seguir descreve como configurar tags para um estágio da API.

Como configurar tags para um estágio de API usando o console do API Gateway

1. Inicie uma sessão no console do API Gateway.
2. Escolha uma API existente ou crie uma nova API que inclua recursos, métodos e as integrações correspondentes.
3. Escolha um estágio ou implante a API em um novo estágio.
4. No painel de navegação principal, selecione Estágios.
5. Escolha a guia Tags. Talvez seja necessário selecionar o botão de seta para a direita para mostrar a guia.
6. Selecione Gerenciar tags.
7. No Editor de tags, selecione Adicionar tag. Insira uma chave de tag (por exemplo, Department) na coluna Key (Chave) e insira um valor de tag (por exemplo, Sales) na coluna Value (Valor). Selecione Salvar para salvar a tag.
8. Se necessário, repita a etapa 5 para adicionar mais tags ao estágio da API. O número máximo de tags por estágio é 50.

9. Para remover uma tag do estágio, selecione Remover.
10. Se a API tiver sido implantada anteriormente no console do API Gateway, será necessário reimplantá-la para que as alterações entrem em vigor.

### Configurar tags para um estágio da API usando a AWS CLI

Você pode configurar tags para um estágio da API usando a AWS CLI por meio do comando [create-stage](#) ou [tag-resource](#). Você pode excluir uma ou mais tags de um estágio da API usando o comando [untag-resource](#).

O comando [create-stage](#) indicado abaixo adiciona uma tag ao criar um estágio test:

```
aws apigateway create-stage --rest-api-id abc1234 --stage-name test --description
'Testing stage' --deployment-id efg456 --tag Department=Sales
```

O comando [tag-resource](#) indicado abaixo adiciona uma tag a um estágio prod:

```
aws apigateway tag-resource --resource-arn arn:aws:apigateway:us-east-2::/
restapis/abc123/stages/prod --tags Department=Sales
```

O comando [untag-resource](#) indicado abaixo remove a tag Department=Sales do estágio test:

```
aws apigateway untag-resource --resource-arn arn:aws:apigateway:us-east-2::/
restapis/abc123/stages/test --tag-keys Department
```

### Configurar tags para um estágio de API usando a API REST do API Gateway

Você pode configurar tags para um estágio de API usando a API REST do API Gateway, com uma das seguintes ações:

- Chame [tags:tag](#) para marcar um estágio de API.
- Chame [tags:untag](#) para excluir uma ou mais tags de um estágio de API.
- Chame [stage:create](#) para adicionar uma ou mais tags a um estágio de API que você estiver criando.

Você também pode chamar [tags:get](#) para descrever as tags em um estágio de API.

## Marcar um estágio da API

Após implantar uma API (m5zr3vnks7) em um estágio (test), marque o estágio chamando [tags:tag](#). O Nome do recurso da Amazon (ARN) do estágio requerido (arn:aws:apigateway:us-east-1::/restapis/m5zr3vnks7/stages/test) deve ser um URL codificado. (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"
 }
}
```

Você também pode usar a solicitação anterior para atualizar uma tag existente para um novo valor.

Você pode adicionar tags a um estágio ao chamar [stage:create](#) para criar o estágio:

```
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"
 }
}
```



## Desmarcar um estágio da API

Para remover a tag Department do estágio, chame [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 remover mais de uma tag, use uma lista de chaves de tags separadas por vírgulas na expressão da consulta; por exemplo, `?tagKeys=Department,Division,....`

## Descrever tags para o estágio da API

Para descrever as tags existentes em um estágio, chame [tags:get](#):

```
GET /tags/arn%3Aaws%3Aapigateway%3Aus-east-1%3A%3A%2Frestapis%2Fm5zr3vnks7%2Fstages%2Ftags
Host: apigateway.us-east-1.amazonaws.com
Authorization: ...
```

A resposta correta é semelhante ao seguinte:

```
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"
}
}
```

## Usar variáveis de estágio para uma API REST no API Gateway

Variáveis de estágio são pares de chave-valor que você pode definir como atributos de configuração associados a um estágio de implantação de uma API REST. Elas atuam como variáveis de ambiente e podem ser usadas em seus modelos de configuração e mapeamento de API. Com estágios de implantação no API Gateway, é possível gerenciar vários estágios de versão para cada API e usar variáveis de estágio. Você pode configurar um estágio de implantação da API para interagir com diferentes endpoints de backend.

As variáveis de estágio não se destinam a ser usadas para dados confidenciais, como credenciais. Para transmitir dados confidenciais para integrações, use um autorizador do AWS Lambda. Você pode passar dados confidenciais para integrações na saída do autorizador do Lambda. Para saber mais, consulte [the section called “Saída de um autorizador do Lambda para o API Gateway”](#).

## Casos de uso de variáveis de estágio

Veja a seguir casos de uso para suas variáveis de estágio.

### Especificar um endpoint de backend diferente

Sua API pode transmitir uma solicitação GET como um proxy HTTP ao host da web de backend. É possível usar uma variável de estágio de forma que, quando chamadores da API invocarem seu endpoint de produção, o API Gateway chame `example.com`. Então, quando os chamadores da API invocam o estágio beta, o API Gateway chama um host da web diferente, como `beta.example.com`. Da mesma forma, variáveis de estágio podem ser usadas para especificar um nome de função AWS Lambda para cada estágio da sua API. Não é possível usar uma variável de estágio para definir um endpoint de integração diferente, como apontar a solicitação GET para uma integração de proxy HTTP em um estágio e uma integração de proxy do Lambda em outro estágio.

Ao especificar um nome de função do Lambda como um valor de variável de estágio, você deve configurar as permissões nessa função do Lambda manualmente. Quando você especifica uma função do Lambda no console do API Gateway, um comando AWS CLI é exibido para configurar as permissões adequadas. Também é possível usar o comando AWS CLI a seguir para fazer isso.

```
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
```

## Transmitir informações usando modelos de mapeamento

É possível acessar variáveis de estágio em modelos de mapeamento ou transmitir parâmetros de configuração ao backend HTTP ou AWS Lambda. Por exemplo, talvez você queira reutilizar a mesma função do Lambda para vários estágios na sua API, mas a função deve ler dados de outra tabela do Amazon DynamoDB, dependendo do estágio. Nos modelos de mapeamento que geram a solicitação para a função do Lambda, você pode usar variáveis de estágio para transmitir o nome da tabela ao Lambda.

Para usar uma variável de estágio, primeiro você deve configurar uma variável de estágio e, depois, atribuir a ela um valor. Por exemplo, para personalizar o endpoint de integração HTTP, primeiro crie a variável de estágio `url` e, depois, na solicitação de integração da sua API, insira o valor da variável de estágio, `http://${stageVariables.url}`. Esse valor instrui o API Gateway a substituir sua variável de estágio `${}` em tempo de execução, dependendo de qual estágio sua API está executando. Para obter mais informações, consulte [the section called “Configurar variáveis de estágio para APIs REST no API Gateway”](#).

## Configurar variáveis de estágio para APIs REST no API Gateway

Esta seção mostra como configurar diversas variáveis de estágio para dois estágios de implantação de uma API de exemplo usando o console do Amazon API Gateway. Para entender como usar variáveis de estágio no API Gateway, recomendamos seguir todos os procedimentos desta seção.

### Pré-requisitos

Antes de começar, certifique-se de que os seguintes pré-requisitos são atendidos:

- Você deve ter uma API disponível no API Gateway. Siga as instruções em [Desenvolver APIs REST no API Gateway](#).
- Você deve ter implantado a API pelo menos uma vez. Siga as instruções em [Implantar APIs REST no API Gateway](#).
- Você deve ter criado o primeiro estágio para uma API implantada. Siga as instruções em [Criar um novo estágio](#).

Invocar um endpoint HTTP por meio de uma API com uma variável de estágio

Este procedimento descreve como criar uma variável de estágio para um endpoint HTTP e dois estágios para sua API. Além disso, você precisa criar as variáveis de estágio `url`, `stageName` e `function` que são usadas nos procedimentos a seguir nesta seção.

Como invocar um endpoint HTTP por meio de uma API com uma variável de estágio

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Crie uma API e, depois, crie um método GET no recurso raiz da API. Defina o tipo de integração como HTTP e o URL do endpoint como **`http://${stageVariables.url}`**.
3. Implante a API em um novo estágio denominado **beta**.
4. No painel de navegação principal, selecione Estágios e, depois, escolha um estágio beta.
5. Na guia Variáveis de estágio, selecione Editar.
6. Selecione Adicionar variável de estágio.
7. Em Nome, digite **url**. Em valor, insira **`httpbin.org/get`**.
8. Selecione Adicionar variável de estágio e, depois, faça o seguinte:  
  
Em Nome, digite **stageName**. Em valor, insira **beta**.
9. Selecione Adicionar variável de estágio e, depois, faça o seguinte:  
  
Em Nome, digite **function**. Em valor, insira **HelloWorld**.
10. Escolha Salvar.
11. Agora crie um segundo estágio. No painel de navegação Estágios, selecione Criar estágio. Em Stage name (Nome do estágio), insira **prod**. Selecione uma implantação recente em Implantação e escolha Criar estágio.
12. Assim como no estágio beta, defina as mesmas três variáveis de estágio (`url`, `stageName` e `function`) como valores diferentes (**`petstore-demo-endpoint.execute-api.com/petstore/pets`**, **prod** e **HelloEveryone**), respectivamente.
13. No painel de navegação Stages (Estágios), escolha o estágio beta. Em Detalhes do estágio, selecione o ícone de cópia para copiar o URL de invocação da API e insira o URL de invocação da API em um navegador da web. Isso inicia a solicitação GET de estágio beta no recurso raiz da API.

 Note

O link Invocar URL aponta para o recurso raiz da API em seu estágio beta. Digitar o URL em um navegador da web chama o método GET de estágio beta no recurso raiz. Se houver métodos definidos em recursos filho, e não no próprio recurso raiz, inserir o URL em um navegador da web gerará uma resposta de erro `{"message": "Missing Authentication Token"}`. Nesse caso, você deve acrescentar o nome de um recurso filho específico ao link Invoke URL (Invocar URL).

14. A resposta que você obteve da solicitação GET de estágio beta é mostrada a seguir. Você também pode verificar o resultado usando um navegador para navegar até `http://httpbin.org/get`. Esse valor foi atribuído à variável `url` no estágio beta. As duas respostas são idênticas.
15. No painel de navegação Stages (Estágios), escolha o nome do estágio prod. Em Detalhes do estágio, selecione o ícone de cópia para copiar o URL de invocação da API e insira o URL de invocação da API em um navegador da web. Isso inicia a solicitação GET de estágio prod no recurso raiz da API.
16. A resposta que você obteve da solicitação GET de estágio prod é mostrada a seguir. Você pode verificar o resultado usando um navegador para navegar para `http://petstore-demo-endpoint.execute-api.com/petstore/pets`. Esse valor foi atribuído à variável `url` no estágio prod. As duas respostas são idênticas.

## Transmitir metadados específicos do estágio para um backend HTTP

Este procedimento descreve como usar um valor de variável de estágio em uma expressão de parâmetro de consulta para transmitir metadados específicos de estágio para um backend HTTP. Usaremos a variável de estágio `stageName` declarada no procedimento anterior.

### Como transmitir metadados específicos do estágio para um backend HTTP

1. No painel de navegação Resource (Recurso), escolha o método GET.

Para adicionar um parâmetro de string de consulta ao URL do método, selecione a guia Solicitação de método e, na seção Configurações de solicitação de método, escolha Editar.

2. Selecione Parâmetros de string de consulta de URL e faça o seguinte:
  - a. Escolha Add query string (Adicionar string de consulta).
  - b. Em Nome, digite **stageName**.

- c. Mantenha Obrigatório e Armazenamento em cache desativados.
3. Escolha Salvar.
4. Escolha a guia Solicitação de integração e, na seção Configurações de solicitação de integração, selecione Editar.
5. Em URL do endpoint, acrescente `?stageName=${stageVariables.stageName}` ao valor do URL definido anteriormente, de forma que todo o URL do endpoint seja **`http://${stageVariables.url}?stageName=${stageVariables.stageName}`**.
6. Selecione Implantar API e o estágio beta.
7. No painel de navegação principal, selecione Estágios. No painel de navegação Stages (Estágios), escolha o estágio beta. Em Detalhes do estágio, selecione o ícone de cópia para copiar o URL de invocação da API e insira o URL de invocação da API em um navegador da web.

 Note

Usamos o estágio beta aqui porque o endpoint HTTP (conforme especificado pela variável `url`, `"http://httpbin.org/get"`) aceita expressões de parâmetro de consulta e retorna-os como o objeto `args` em sua resposta.

8. Você receberá a seguinte resposta. Observe que `beta`, atribuído à variável de estágio `stageName`, é passado para o backend como o 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=1-11111111-1111111111111111"
 },
 "origin": "192.0.2.9",
 "url": "http://httpbin.org/get?stageName=beta"
}
```

## Invocar uma função do Lambda por meio de uma API com uma variável de estágio

Este procedimento descreve como usar uma variável de estágio para chamar uma função do Lambda como um backend da sua API. Você usará a variável de estágio `function` declarada em [Invocar um endpoint HTTP por meio de uma API com uma variável de estágio](#).

Ao definir uma função do Lambda como o valor de uma variável de estágio, use o nome local da função, possivelmente incluindo seu alias ou sua especificação de versão, como em **HelloWorld**, **HelloWorld:1** ou **HelloWorld:alpha**. Não use o ARN da função (por exemplo, **arn:aws:lambda:us-east-1:123456789012:function:HelloWorld**). O console do API Gateway pressupõe o valor da variável de estágio de uma função do Lambda como o nome de função não qualificado e expande a variável de estágio especificada em um ARN.

Como invocar uma função do Lambda por meio de uma API com uma variável de estágio

1. Crie uma função do Lambda chamada **HelloWorld** usando o runtime Node.js padrão. O código deve conter o seguinte:

```
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 obter mais informações sobre como criar uma função do Lambda, consulte [Conceitos básicos do console da API REST](#).

2. No painel Recursos, selecione Criar recurso e faça o seguinte:
  - a. Em Caminho do recurso, selecione/.
  - b. Em Resource Name (Nome do recurso), insira **lambdav1**.
  - c. Selecione Criar recurso.
3. Selecione o recurso /lambdav1 e escolha Criar método.

Faça o seguinte:

- a. Em Tipo de método, selecione GET.
- b. Em Tipo de integração, selecione Função do Lambda.

- c. Mantenha a opção Integração do proxy do Lambda desativada.
- d. Em Lambda function (Função do Lambda), insira `${stageVariables.function}`.

**Lambda function**

Provide the Lambda function name or alias. You can also provide an ARN from another account.

us-east-1 ▼

🔍

`${stageVariables.function}`

✕

**Tip**

Quando for solicitado para Adicionar comando de permissão, copie o comando [add-permission](#). Execute o comando em cada função do Lambda que será atribuída à variável de estágio `function`. Por exemplo, se o valor `$stageVariables.function` for `HelloWorld`, execute o seguinte comando da 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
```

Se isso não for feito, uma resposta `500 Internal Server Error` será gerada ao invocar o método. Substitua `${stageVariables.function}` pelo nome da função do Lambda atribuída à variável de estágio.



**Lambda function**  
Provide the Lambda function name or alias. You can also provide an ARN from another account.

us-east-1

 **You defined your Lambda function as a stage variable. Run the following AWS CLI command to ensure you have the appropriate policy for this function. Replace the stage variable in the function-name parameter with the necessary function name.**

▼ Add permission command

```

1 aws lambda add-permission \
2 --function-name "arn:aws:lambda:us-east-1:111122223333:function:${stageVariables.function}" \
3 --source-arn "arn:aws:execute-api:us-east-1:111122223333:abcd1234/*/GET/lambdav1" \
4 --principal apigateway.amazonaws.com \
5 --statement-id abcd-12345-efg \
6 --action lambda:InvokeFunction

```

e. Escolha Criar método.

4. Implante a API nos estágios prod e beta.
5. No painel de navegação principal, selecione Estágios. No painel de navegação Stages (Estágios), escolha o estágio beta. Em Detalhes do estágio, selecione o ícone de cópia para copiar o URL de invocação da API e insira o URL de invocação da API em um navegador da web. Anexe **/lambdav1** ao URL antes de pressionar enter.

Você receberá a seguinte resposta.

"Hello, World! I'm not sure where I'm calling from..."

Transmitir metadados específicos ao estágio para uma função do Lambda por meio de uma variável de estágio

Este procedimento descreve como usar uma variável de estágio para transmitir metadados de configuração específicos de estágio para uma função do Lambda. Você criará um método POST e um modelo de mapeamento de entrada para gerar a carga útil usando a variável de estágio `stageName` declarada anteriormente.

Como transmitir metadados específicos ao estágio para uma função do Lambda por meio de uma variável de estágio

1. Selecione o recurso `/lambdav1` e escolha Criar método.

Faça o seguinte:

- a. Em Tipo de método, selecione POST.
  - b. Em Tipo de integração, selecione Função do Lambda.
  - c. Mantenha a opção Integração do proxy do Lambda desativada.
  - d. Em Lambda function (Função do Lambda), insira `${stageVariables.function}`.
  - e. Quando for solicitado para Adicionar comando de permissão, copie o comando [add-permission](#). Execute o comando em cada função do Lambda que será atribuída à variável de estágio `function`.
  - f. Escolha Criar método.
2. Escolha a guia Solicitação de integração e, na seção Configurações de solicitação de integração, selecione Editar.
  3. Selecione Modelos de mapeamento e, depois, Adicionar modelo de mapeamento.
  4. Em Tipo de conteúdo, insira **application/json**.
  5. Em Corpo do modelo, insira o seguinte modelo:

```
#set($inputRoot = $input.path('$'))
{
 "stageName" : "$stageVariables.stageName"
}
```

 Note

Em um modelo de mapeamento, uma variável de estágio deve ser referenciada entre aspas (como em `"$stageVariables.stageName"` ou `"${stageVariables.stageName}"`). Em outros lugares, ela deve ser referenciada sem aspas (como em `${stageVariables.function}`).

6. Escolha Salvar.
7. Implante a API nos estágios beta e prod.
8. Para usar um cliente da API REST para transmitir metadados específicos do estágio, faça o seguinte:
  - a. No painel de navegação Stages (Estágios), escolha o estágio beta. Em Detalhes do estágio, selecione o ícone de cópia para copiar o URL de invocação da API e, depois, insira o URL

de invocação da API no campo de entrada de um cliente da API REST. Anexe **/lambdav1** antes de enviar a solicitação.

Você receberá a seguinte resposta.

```
"Hello, World! I'm calling from the beta stage."
```

- b. No painel de navegação Estágios, selecione o estágio prod. Em Detalhes do estágio, selecione o ícone de cópia para copiar o URL de invocação da API e, depois, insira o URL de invocação da API no campo de entrada de um cliente da API REST. Anexe **/lambdav1** antes de enviar a solicitação.

Você receberá a seguinte resposta.

```
"Hello, World! I'm calling from the prod stage."
```

9. Para usar o recurso Testar para transmitir metadados específicos do estágio, faça o seguinte:

- a. No painel de navegação Recursos, selecione a guia Testar. Talvez seja necessário selecionar o botão de seta para a direita para mostrar a guia.
- b. Em função, insira **HelloWorld**.
- c. Em stageName, insira **beta**.
- d. Escolha Testar. Não é necessário adicionar um corpo à solicitação POST.

Você receberá a seguinte resposta.

```
"Hello, World! I'm calling from the beta stage."
```

- e. É possível repetir as etapas anteriores para testar o estágio Prod. Em stageName, insira **Prod**.

Você receberá a seguinte resposta.

```
"Hello, World! I'm calling from the prod stage."
```

Referência de variáveis de estágio do API Gateway para APIs REST no API Gateway

Você pode usar variáveis de estágio do API Gateway nos seguintes casos.

## Expressões de mapeamento de parâmetros

Uma variável de estágio pode ser usada em uma expressão de mapeamento de parâmetros para o parâmetro de cabeçalho de solicitação ou resposta de um método de API, sem substituição parcial. No exemplo a seguir, a variável de estágio é referenciada sem o \$ e o delimitador { . . . }.

- `stageVariables.<variable_name>`

## Modelos de mapeamento

Uma variável de estágio pode ser usada em qualquer lugar de um modelo de mapeamento, conforme mostrado nos exemplos a seguir.

- `{ "name" : "$stageVariables.<variable_name>" }`
- `{ "name" : "${stageVariables.<variable_name>}" }`

## URIs de integração HTTP

Uma variável de estágio pode ser usada como parte de um URL de integração HTTP, como mostram os exemplos a seguir:

- Um URI completo sem protocolo – `http://${stageVariables.<variable_name>}`
- Um domínio complet – `http://${stageVariables.<variable_name>}/resource/operation`
- Um subdomíni – `http://${stageVariables.<variable_name>}.example.com/resource/operation`
- Um caminh – `http://example.com/${stageVariables.<variable_name>}/bar`
- Uma string de consult – `http://example.com/foo?q=${stageVariables.<variable_name>}`

## AWSURIs de integração da

Uma variável de estágio pode ser usada como parte de componentes de caminho ou ação de URI da AWS, como mostra o exemplo a seguir.

- `arn:aws:apigateway:<region>:<service>:${stageVariables.<variable_name>}`

## AWSURIs de integração da (funções do Lambda)

Uma variável de estágio pode ser usada no lugar de um nome de função do Lambda, ou de uma versão/alias, como mostram os exemplos a seguir.

- `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 usar uma variável de estágio para uma função do Lambda, a função deve estar na mesma conta que a API. As variáveis de estágio não suportam funções do Lambda entre contas.

## Grupo de usuários do Amazon Cognito

Uma variável de estágio pode ser usada no lugar de um grupo de usuários do Amazon Cognito para um autorizador COGNITO\_USER\_POOLS.

- `arn:aws:cognito-idp:<region>:<account_id>:userpool/${stageVariables.<variable_name>}`

## AWSCredenciais de integração da

Uma variável de estágio pode ser usada como parte do ARN de credencial de usuário/função da AWS, como mostra o exemplo a seguir.

- `arn:aws:iam::<account_id>:${stageVariables.<variable_name>}`

## Configurar uma implantação de versão canary do API Gateway

A [versão canary](#) é uma estratégia de desenvolvimento de software em que uma nova versão de uma API (e outros softwares) é implantada para fins de teste, e a versão base permanece implantada

como uma versão de produção para operações normais no mesmo estágio. Para fins de discussão, nós nos referimos à versão base como uma versão de produção nesta documentação. Embora isso seja razoável, você pode aplicar a versão canary a qualquer versão de não produção para testes.

Em uma implantação da versão canary, o tráfego total da API é separado aleatoriamente em uma versão de produção e uma versão canary com uma proporção pré-configurada. Geralmente, a versão canary recebe uma pequena porcentagem do tráfego da API e a versão de produção obtém o resto. Os recursos de API atualizados são visíveis para o tráfego da API apenas por meio do canary. Você pode ajustar a porcentagem do tráfego do canary para otimizar a cobertura ou a performance do teste.

Ao manter o tráfego canary pequeno e a seleção aleatória, a maioria dos usuários não é afetada negativamente a qualquer momento por potenciais erros na nova versão, e nenhum usuário único é afetado negativamente o tempo todo.

Depois que as métricas de teste aprovarem seus requisitos, você poderá promover a versão canary para a versão de produção e desativar o canary da implantação. Isso disponibiliza os novos recursos na etapa de produção.

## Tópicos

- [Implantação da versão canary no API Gateway](#)
- [Criar uma implantação da versão canary](#)
- [Atualizar uma versão canary](#)
- [Promover uma versão canary](#)
- [Desativar uma versão de canário](#)

## Implantação da versão canary no API Gateway

No API Gateway, uma implantação da versão canary usa o estágio de implantação para a versão de produção da versão base de uma API e atribui ao estágio uma versão canary para as novas versões, em relação à versão base, da API. O estágio é associado à implantação inicial e o canary às implantações subsequentes. No início, tanto o estágio quanto o canary apontam para a mesma versão da API. Utilizamos estágio e versão de produção como sinônimos e usamos canary e versão canary também como sinônimos nesta seção.

Para implantar uma API com uma versão canary, crie uma implantação da versão canary adicionando [configurações do canary](#) ao [estágio](#) de uma [implantação](#) normal. As configurações do

canary descrevem a versão canary subjacente e o estágio representa a versão de produção da API nesta implantação. Para adicionar as configurações do canary, defina `canarySettings` no estágio da implantação e especifique o seguinte:

- Um ID de implantação inicialmente idêntico ao ID da implantação da versão base definido no estágio.
- Uma [porcentagem do tráfego da API](#), entre 0,0 e 100,0 (ambos incluídos), para a versão canary.
- [Variáveis de estágio para a versão canary](#) que podem substituir as variáveis de estágio da versão de produção.
- O [uso do cache do estágio](#) para solicitações do canary, se o [useStageCache](#) estiver definido e o armazenamento em cache da API estiver ativado no estágio.

Depois que uma versão canary é ativada, o estágio de implantação não pode ser associado a outra implantação da versão não canary até que a versão canary seja desativada e as suas configurações sejam removidas do estágio.

Quando você ativa o log de execução da API, a versão canary possui seus próprios registros e métricas geradas para todas as solicitações do canary. Eles são relatados a um grupo de logs do CloudWatch Logs do estágio de produção e a um grupo de logs do CloudWatch Logs específico ao canary. O mesmo se aplica ao registro de acesso. Os registros específicos do canary separados são úteis para validar novas alterações na API e decidir se aceitam as alterações e promovem a versão canary para o estágio de produção ou se descartam as alterações e reverterem a versão canary do estágio de produção.

O grupo de logs de execução do estágio de produção é chamado `API-Gateway-Execution-Logs/{rest-api-id}/{stage-name}` e o grupo de logs de execução da versão canary é chamado `API-Gateway-Execution-Logs/{rest-api-id}/{stage-name}/Canary`. Para o registro de acesso, você deve criar um novo grupo de logs ou escolher um existente. O nome do grupo de logs de acesso da versão canary tem o sufixo `/Canary` anexado ao nome do grupo de logs selecionado.

Uma versão canary pode usar o cache do estágio, se ativado, para armazenar respostas e usar entradas em cache para retornar resultados para as próximas solicitações canary, dentro de um período de TTL (Time-to-Live) pré-configurado.

Em uma implantação da versão canary, a versão de produção e a versão canary da API podem ser associadas a mesma versão ou a versões diferentes. Quando elas são associadas a versões

diferentes, as respostas para produção e solicitações do canary são armazenadas em cache separadamente e o cache do estágio retorna resultados correspondentes para a produção e solicitações do canary. Quando a versão de produção e a versão canary são associadas a mesma implantação, o cache do estágio usa uma única chave de cache para ambos os tipos de solicitações e retorna a mesma resposta para as mesmas solicitações da versão de produção e versão canary.

### Criar uma implantação da versão canary

Crie uma implantação da versão canary ao implantar a API com [configurações do canary](#) como uma entrada adicional para a operação de [criação de implantação](#).

Você também pode criar uma implantação da versão canary a partir de uma implantação não canary existente fazendo uma solicitação [stage:update](#) para adicionar as configurações do canary no estágio.

Ao criar uma implantação da versão não canary, você pode especificar um nome de estágio não existente. O API Gateway criará um se o estágio especificado não existir. No entanto, você não pode especificar um nome de estágio não existente ao criar uma implantação da versão canary. Você receberá um erro e o API Gateway não criará qualquer implantação da versão canary.

Você pode criar uma implantação do lançamento canary no API Gateway usando o console do API Gateway, a CLI AWS CLI ou um SDK da AWS.

### Tópicos

- [Criar uma implantação canary usando o console do API Gateway](#)
- [Criar uma implantação do canary usando a AWS CLI](#)

### Criar uma implantação canary usando o console do API Gateway

Para usar o console do API Gateway para criar uma implantação da versão canary, siga as instruções abaixo:

Para criar a implantação da versão canary inicial

1. Inicie uma sessão no console do API Gateway.
2. Selecione uma API REST existente ou crie uma API REST.
3. No painel de navegação principal, selecione Recursos e, depois, Implantar API. Siga as instruções na tela em Deploy API para implantar a API em um novo estágio.



Até agora, você implantou a API em um estágio da versão de produção. Em seguida, configure as configurações do canary no estágio e, se necessário, também ative o armazenamento em cache, defina variáveis de estágio ou configure a execução da API ou registros de acesso.

4. Para habilitar o armazenamento em cache da API ou associar uma ACL da web do AWS WAF ao estágio, na seção Detalhes do estágio, escolha Editar. Para ter mais informações, consulte [the section called “Configurações de cache”](#) ou [the section called “Associar uma ACL da web do AWS WAF a um estágio da API do API Gateway usando o console do API Gateway”](#).
5. Para configurar a execução ou o registro em log do acesso, na guia Logs e rastreamento, selecione Editar e siga as instruções na tela. Para obter mais informações, consulte [Configurar o registro em log do CloudWatch para APIs REST no API Gateway](#).
6. Para definir variáveis de estágio, escolha a guia Variáveis de estágio e siga as instruções na tela para adicionar ou modificar as variáveis de estágio. Para obter mais informações, consulte [the section called “Usar variáveis de estágio”](#).
7. Escolha a guia Canário e selecione Criar canário. Talvez seja necessário escolher o botão de seta para a direita para mostrar a guia Canário.
8. Em Configurações de canário, em Canário, insira a porcentagem de solicitações a serem desviadas para o canário.
9. Se desejar, selecione Cache de estágio para ativar o armazenamento em cache da versão de canário. O cache não estará disponível para a versão canary até o armazenamento em cache da API ser ativado.
10. Para substituir variáveis de estágio existentes, em Substituição do canário, insira um novo valor de variável de estágio.

Depois que a versão canary for inicializada no estágio de implantação, altere a API e teste as alterações. Reimplante a API no mesmo estágio para que a versão atualizada e a versão base possam ser acessadas através do mesmo estágio. As etapas a seguir descrevem como fazer isso:

Para implantar a última versão da API em um canary

1. Com cada atualização da API, selecione Implantar API.
2. Em Implantar API, escolha o estágio que contém um canário na lista suspensa Estágio de implantação.
3. (Opcional) Insira uma descrição em Descrição da implantação.
4. Escolha Deploy para enviar a última versão da API para a versão canary.

5. Se desejar, reconfigure as configurações do estágio, registros ou as configurações do canary, conforme descrito em [Para criar a implantação da versão canary inicial](#).

Como resultado, a versão canary aponta para a versão mais recente enquanto a versão de produção ainda aponta para a versão inicial da API. As [canarySettings](#) agora têm um novo valor `deploymentId`, enquanto o estágio ainda mantém o valor inicial `deploymentId`. Enquanto isso, o console chama [stage:update](#).

Criar uma implantação do canary usando a AWS CLI

Como criar uma implantação canário para um novo estágio

1. Use o comando [create-deployment](#) indicado abaixo para criar uma implantação com duas variáveis de estágio, mas sem um canário:

```
aws apigateway create-deployment \
 --variables sv0=val0,sv1=val1 \
 --rest-api-id abcd1234 \
 --stage-name 'prod'
```

A saída será exibida da seguinte forma:

```
{
 "id": "du4ot1",
 "createdDate": 1511379050
}
```

2. Use o comando [create-deployment](#) indicado abaixo para criar uma implantação canário no estágio prod:

```
aws apigateway create-deployment \
 --rest-api-id abcd1234 \
 --canary-settings
 percentTraffic=10.5,stageVariableOverrides={sv1='val2',sv2='val3'},useStageCache=false
 \
 --stage-name 'prod'
```

A saída será exibida da seguinte forma:

```
{
```

```
"id": "a6rox0",
"createdDate": 1511379433
}
```

O id de implantação resultante identifica a versão de teste da API para a versão canary. Como resultado, o estágio associado é ativado para canary.

3. (Opcional) Use o comando [get-stage](#) indicado abaixo para visualizar a representação do estágio:

```
aws apigateway get-stage --rest-api-id acbd1234 --stage-name prod
```

A seguir está uma representação do Stage como resultado de um 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": {}
}
```

Neste exemplo, a versão base da API usará as variáveis de estágio do {"sv0": val0", "sv1": val1"}, enquanto a versão de teste usa as variáveis de estágio do {"sv1": val2", "sv2": val3"}. A versão de produção e a versão canary usam a mesma variável de estágio do sv1, mas com valores diferentes, val1 e val2, respectivamente. A variável de estágio do

sv0 é usada somente na versão de produção e a variável de estágio do sv2 é usada somente na versão canary.

Você também pode criar uma implantação da versão canário com base em uma implantação normal atualizando o estágio para ativar um canário.

Como criar uma implantação de versão canário com base em uma implantação existente

1. Use o comando [create-deployment](#) para criar uma implantação sem um canário:

```
aws apigateway create-deployment \
 --variables sv0=val0,sv1=val1 \
 --rest-api-id abcd1234 \
 --stage-name 'beta'
```

2. Use o comando [update-stage](#) para atualizar o estágio e habilitar um canário:

```
aws apigateway update-stage \
 --rest-api-id abcd1234 \
 --stage-name 'beta' \
 --patch-operations ' [{
 "op": "replace",
 "value": "0.0",
 "path": "/canarySettings/percentTraffic"
 }, {
 "op": "copy",
 "from": "/canarySettings/stageVariable0overrides",
 "path": "/variables"
 }, {
 "op": "copy",
 "from": "/canarySettings/deploymentId",
 "path": "/deploymentId"
 }]'
```

A saída será exibida da seguinte forma:

```
{
 "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 acabamos de habilitar um canário em uma versão existente da API, a versão de produção (Stage) e a versão canário (canarySettings) apontam para a mesma implantação. Depois de alterar a API e implantá-la nesse estágio novamente, a nova versão ficará na versão canary, enquanto a versão base permanecerá na versão de produção. Isso se manifesta na evolução do estágio quando o deploymentId na versão canary é atualizado para a nova implantação do id e o deploymentId na versão de produção permanece inalterado.

## Atualizar uma versão canary

Após uma versão canary ser implantada, ajuste a porcentagem do tráfego canary ou ative ou desative o uso de um cache do estágio para otimizar a performance. Você também pode modificar as variáveis de estágio usadas na versão canary quando o contexto de execução é atualizado. Para fazer essas atualizações, chame a operação [stage:update](#) com novos valores em [canarySettings](#).

Você pode atualizar um lançamento canary usando o console do API Gateway, o comando da AWS CLI [update-stage](#) ou um SDK da AWS.

## Tópicos

- [Atualizar uma versão canary usando o console do API Gateway](#)
- [Atualizar uma versão canary usando a AWS CLI](#)

## Atualizar uma versão canary usando o console do API Gateway

Para usar o console do API Gateway para atualizar as configurações do canary existentes em um estágio, faça o seguinte:

Como atualizar as configurações existentes do canário

1. Faça login no console do API Gateway e selecione uma API REST existente.
2. No painel de navegação principal, selecione Estágios e escolha um estágio existente.
3. Selecione a guia Canário e Editar. Talvez seja necessário escolher o botão de seta para a direita para mostrar a guia Canário.
4. Atualize Solicitar distribuição aumentando ou diminuindo a porcentagem entre 0,0 e 100,0.
5. Marque ou desmarque a caixa de seleção Cache de estágio.
6. Adicione, remova ou modifique as Variáveis de estágio de canário.
7. Escolha Salvar.

## Atualizar uma versão canary usando a AWS CLI

Para atualizar um canário por meio da AWS CLI, use o comando [update-stage](#) e modifique a operação de patch para cada parâmetro do canário.

O comando [update-stage](#) indicado abaixo atualiza se o canário usar o cache de estágio:

```
aws apigateway update-stage \
 --rest-api-id {rest-api-id} \
 --stage-name '{stage-name}' \
 --patch-operations op=replace,path=/canarySettings/useStageCache,value=true
```

O comando [update-stage](#) indicado abaixo atualiza a porcentagem de tráfego do canário:

```
aws apigateway update-stage \
 --rest-api-id {rest-api-id} \
 --stage-name '{stage-name}' \
 --patch-operations op=replace,path=/canarySettings/percentTraffic,value=25.0
```

O comando [update-stage](#) a seguir atualiza as variáveis de estágio. O exemplo mostra como criar uma variável de estágio chamada newVar, substituir a variável de estágio var2 e remover a variável de estágio var1:

```
aws apigateway update-stage \
 --rest-api-id {rest-api-id} \
 --stage-name '{stage-name}' \
 --patch-operations ' [{
 "op": "replace",
 "path": "/canarySettings/stageVariable0overrides/newVar",
 "value": "newVal"
 }, {
 "op": "replace",
 "path": "/canarySettings/stageVariable0overrides/var2",
 "value": "val4"
 }, {
 "op": "remove",
 "path": "/canarySettings/stageVariable0overrides/var1"
 }]'
```

Você pode atualizar todas acima combinando as operações em um único 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/stageVariable0overrides/var1"
 }, {
 "op": "replace",
 "path": "/canarySettings/stageVariable0overrides/newVar",
 "value": "newVal"
 }, {
 "op": "replace",
 "path": "/canarySettings/stageVariable0overrides/val2",
 "value": "val4"
 }]'
```

## Promover uma versão canary

Ao promover uma versão canário, ela substitui as configurações atuais do estágio. A promoção de uma versão canary não desativa o canary no estágio. Para desativar um canary, você deve remover as configurações do canary do estágio. Para promover uma versão canário, realize o procedimento a seguir.

- Redefina o [ID de implantação](#) do estágio com as configurações do [ID de implantação](#) do canary. Isso atualiza o snapshot da API do estágio com o snapshot do canary, tornando a versão de teste também a versão de produção.
- Atualize as variáveis de estágio com as variáveis de estágio do canary, se houver alguma. Isso atualiza o contexto de execução da API do estágio com o do canary. Sem essa atualização, a nova versão da API pode produzir resultados inesperados se a versão de teste usar diferentes variáveis de estágio ou valores diferentes de variáveis de estágio existentes.
- Defina a porcentagem do tráfego do canary para 0,0 %.

## Tópicos

- [Promover uma versão canary usando o console do API Gateway](#)
- [Promover uma versão canary usando a AWS CLI](#)

## Promover uma versão canary usando o console do API Gateway

Para usar o console do API Gateway para promover uma implantação da versão canary, faça o seguinte:

### Como promover uma implantação da versão de canário

1. Inicie uma sessão no console do API Gateway e escolha uma API existente no painel de navegação principal.
2. No painel de navegação principal, selecione Estágios e escolha um estágio existente.
3. Selecione a guia Canário.
4. Escolha Promover canário.
5. Confirme as alterações a serem realizadas e selecione Promover canário.



Após a promoção, a versão de produção faz referência à mesma versão da API (deploymentId) que a versão canary. Você pode verificar isso usando a AWS CLI. Por exemplo, consulte [the section called “Promover uma versão canary usando a AWS CLI”](#).

## Promover uma versão canary usando a AWS CLI

Para promover a versão canary para a versão de produção usando os comandos da AWS CLI, chame o comando `update-stage` para copiar o `deploymentId` associado ao canary no `deploymentId` associado ao estágio, para redefinir a porcentagem de tráfego do canary para zero (0.0) e para copiar qualquer variável de estágio vinculada ao canary nas variáveis vinculadas ao estágio correspondentes.

Suponha que tenhamos uma implantação da versão canary, descrita por um estágio semelhante ao seguinte:

```
{
 "_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": "nfcx0x",
 "lastUpdatedDate": "2017-11-22T00:54:28Z",
 "methodSettings": {
 ...
 },
 "stageName": "prod",
 "variables": {
 "sv1": "val1"
 }
}
```

```
}
```

Use o comando [update-stage](#) indicado abaixo para promover o canário:

```
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/stageVariable0overrides",
 "path": "/variables"
 }, {
 "op": "copy",
 "from": "/canarySettings/deploymentId",
 "path": "/deploymentId"
 }]'
```

A saída será exibida da seguinte forma:

```
{
 "_links": {
 ...
 },
 "accessLogSettings": {
 ...
 },
 "cacheClusterEnabled": false,
 "cacheClusterStatus": "NOT_AVAILABLE",
 "canarySettings": {
 "deploymentId": "eh1sby",
 "useStageCache": false,
 "stageVariable0overrides": {
 "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"
}
}
```

Promover uma versão canário ao estágio não a desabilita, e a implantação continua sendo uma implantação da versão canário. Para torná-la uma implantação da versão de produção regular, você deve desabilitar as configurações do canary. Para obter mais informações sobre como desabilitar uma implantação da versão canary, consulte [the section called “Desativar uma versão de canário”](#).

### Desativar uma versão de canário

Para desativar uma implantação da versão de canário, defina [canarySettings](#) como nulo para removê-la do estágio.

Você pode desativar uma implantação do lançamento canary usando o console do API Gateway, a AWS CLI ou um SDK da AWS.

### Tópicos

- [Desativar uma versão de canário usando o console do API Gateway](#)
- [Desativar uma versão de canário usando a AWS CLI](#)

### Desativar uma versão de canário usando o console do API Gateway

Para usar o console do API Gateway para desativar uma implantação da versão de canário, siga as seguintes etapas:

#### Como desativar uma implantação da versão de canário

1. Faça login no console do API Gateway e escolha uma API existente no painel de navegação principal.
2. No painel de navegação principal, selecione Estágios e escolha um estágio existente.
3. Selecione a guia Canário.

4. Escolha Excluir.
5. Confirme se você deseja excluir o canary escolhendo Delete.

Como resultado, a propriedade [canarySettings](#) se torna null e é removida do [estágio](#) da implantação. Você pode verificar isso usando a AWS CLI. Por exemplo, consulte [the section called “Desativar uma versão de canário usando a AWS CLI”](#).

### Desativar uma versão de canário usando a AWS CLI

O comando [update-stage](#) indicado abaixo desativa a implantação da versão canário:

```
aws apigateway update-stage \
 --rest-api-id abcd1234 \
 --stage-name canary \
 --patch-operations '[{"op":"remove", "path":"/canarySettings"}]'
```

A saída será exibida como a seguir:

```
{
 "stageName": "prod",
 "accessLogSettings": {
 ...
 },
 "cacheClusterEnabled": false,
 "cacheClusterStatus": "NOT_AVAILABLE",
 "deploymentId": "nfcn0x",
 "lastUpdatedDate": 1511309280,
 "createdDate": 1511152939,
 "methodSettings": {
 ...
 }
}
```

Como mostrado na saída, a propriedade [canarySettings](#) não está mais presente no [estágio](#) de uma implantação desabilitada para canário.

## Atualizações das APIs REST que exigem reimplantação

A manutenção de uma API consiste em visualizar, atualizar e excluir as configurações de API existentes. Você pode manter uma API usando o console do API Gateway, a AWS CLI, o

CloudFormation, um SDK ou a API REST do API Gateway. A atualização de uma API envolve modificar determinadas propriedades de recursos ou configurações da API. Atualizações de recursos exigem a reimplantação da API, enquanto as atualizações de configuração, não.

A tabela a seguir descreve os recursos da API que exigem a reimplantação da sua API quando são atualizados.

| Recurso                                   | Observações                                                                                                                                         |
|-------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">ApiKey</a>                    | Para propriedades aplicáveis e operações compatíveis, consulte <a href="#">apikey:update</a> . A atualização exige reimplantar a API.               |
| <a href="#">Autorizador</a>               | Para propriedades aplicáveis e operações compatíveis, consulte <a href="#">authorize:update</a> . A atualização exige reimplantar a API.            |
| <a href="#">disableExecuteApiEndpoint</a> | A atualização requer a modificação de qualquer estágio da API, como a reimplantação da API em um estágio.                                           |
| <a href="#">DocumentationPart</a>         | Para propriedades aplicáveis e operações compatíveis, consulte <a href="#">documentationpart:update</a> . A atualização exige reimplantar a API.    |
| <a href="#">DocumentationVersion</a>      | Para propriedades aplicáveis e operações compatíveis, consulte <a href="#">documentationversion:update</a> . A atualização exige reimplantar a API. |
| <a href="#">GatewayResponse</a>           | Para propriedades aplicáveis e operações compatíveis, consulte <a href="#">gatewayresponse:update</a> . A atualização exige reimplantar a API.      |
| <a href="#">Integration</a>               | Para propriedades aplicáveis e operações compatíveis, consulte <a href="#">integration:update</a> . A atualização exige reimplantar a API.          |
| <a href="#">IntegrationResponse</a>       | Para propriedades aplicáveis e operações compatíveis, consulte <a href="#">integrationresponse:update</a> . A atualização exige reimplantar a API.  |
| <a href="#">Método</a>                    | Para propriedades aplicáveis e operações compatíveis, consulte <a href="#">method:update</a> . A atualização exige reimplantar a API.               |
| <a href="#">MethodResponse</a>            | Para propriedades aplicáveis e operações compatíveis, consulte <a href="#">methodresponse:update</a> . A atualização exige reimplantar a API.       |

| Recurso                          | Observações                                                                                                                                                                                |
|----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">Modelo</a>           | Para propriedades aplicáveis e operações compatíveis, consulte <a href="#">model:update</a> . A atualização exige reimplantar a API.                                                       |
| <a href="#">RequestValidator</a> | Para propriedades aplicáveis e operações compatíveis, consulte <a href="#">requestvalidator:update</a> . A atualização exige reimplantar a API.                                            |
| <a href="#">Recurso</a>          | Para propriedades aplicáveis e operações compatíveis, consulte <a href="#">resource:update</a> . A atualização exige reimplantar a API.                                                    |
| <a href="#">RestApi</a>          | Para propriedades aplicáveis e operações compatíveis, consulte <a href="#">restapi:update</a> . A atualização exige reimplantar a API. Isso inclui a modificação de políticas de recursos. |
| <a href="#">VpcLink</a>          | Para propriedades aplicáveis e operações compatíveis, consulte <a href="#">vpclink:update</a> . A atualização exige reimplantar a API.                                                     |

A tabela a seguir descreve as configurações de API que não exigem a reimplantação da API quando são atualizadas.

| Configuração                        | Observações                                                                                                                                        |
|-------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">Conta da</a>            | Para propriedades aplicáveis e operações compatíveis, consulte <a href="#">account:update</a> . A atualização não exige reimplantar a API.         |
| <a href="#">Implantação</a>         | Para propriedades aplicáveis e operações compatíveis, consulte <a href="#">deployment:update</a> .                                                 |
| <a href="#">DomainName</a>          | Para propriedades aplicáveis e operações compatíveis, consulte <a href="#">domainname:update</a> . A atualização não exige reimplantar a API.      |
| <a href="#">BasePathMapping</a>     | Para propriedades aplicáveis e operações compatíveis, consulte <a href="#">basepathmapping:update</a> . A atualização não exige reimplantar a API. |
| <a href="#">Tipo de endereço IP</a> | A atualização não exige reimplantar a API.                                                                                                         |

| Configuração              | Observações                                                                                                                                  |
|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">Estágio</a>   | Para propriedades aplicáveis e operações compatíveis, consulte <a href="#">stage:update</a> . A atualização não exige reimplantar a API.     |
| <a href="#">Usage</a>     | Para propriedades aplicáveis e operações compatíveis, consulte <a href="#">usage:update</a> . A atualização não exige reimplantar a API.     |
| <a href="#">UsagePlan</a> | Para propriedades aplicáveis e operações compatíveis, consulte <a href="#">usageplan:update</a> . A atualização não exige reimplantar a API. |

## Nome de domínio personalizado para APIs REST públicas no API Gateway

Os nomes de domínio personalizados são URLs mais simples e intuitivos que você pode fornecer aos usuários da API.

Após a implantação da sua API, você e seus clientes podem invocar essa API usando o URL de base padrão com o seguinte formato:

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

em que *api-id* é gerado pelo API Gateway, *region* é a região da AWS e *stage* é especificado por você ao implantar a API.

A parte do nome de host do URL, *api-id*.execute-api.*region*.amazonaws.com, refere-se a um endpoint de API. O endpoint de API padrão é gerado aleatoriamente, difícil de lembrar e não é simples de usar.

Com nomes de domínio personalizados, você pode configurar o nome de host da API e escolher um caminho base (por exemplo, myservice) para mapear o URL alternativo para sua API. Por exemplo, um URL de base de API mais amigável pode se tornar:

```
https://api.example.com/myservice
```

### Note

Para ter informações sobre nomes de domínio personalizados para APIs privadas, consulte [the section called “Nomes de domínio personalizados para APIs privadas”](#).

## Considerações

As considerações a seguir podem afetar o uso de um nome de domínio personalizado.

- É possível desabilitar o endpoint padrão para a API. Os clientes ainda podem se conectar ao endpoint padrão, mas receberão um código de status 403 Forbidden.
- Um nome de domínio regional personalizado pode ser associado a APIs REST e APIs HTTP. É possível usar as [APIs do API Gateway versão 2](#) para criar e gerenciar nomes de domínio regionais personalizados para APIs REST.
- Um nome de domínio personalizado deve ser exclusivo em uma região em todas as contas da AWS.
- Você pode migrar o nome de domínio personalizado entre endpoints otimizados para borda e regionais, mas não é possível migrar um domínio personalizado público para um nome de domínio personalizado privado.
- É necessário criar ou atualizar o registro de recursos do provedor DNS para ser mapeado ao endpoint da API. Sem esse mapeamento, as solicitações de API que forem direcionadas para o nome de domínio personalizado não conseguirão acessar o API Gateway.
- É possível comportar um número quase infinito de nomes de domínio sem exceder a cota padrão usando um certificado curinga. Para obter mais informações, consulte [the section called “Nomes de domínio personalizados curinga”](#).
- É possível escolher uma política de segurança para o nome de domínio personalizado. Para obter mais informações, consulte [the section called “Escolher uma política de segurança”](#).
- Para configurar mapeamentos de API com vários níveis, é necessário usar um nome de domínio regional personalizado e a política de segurança do TLS 1.2.

## Pré-requisitos referentes a nomes de domínio personalizados

Veja a seguir os pré-requisitos para criar um nome de domínio personalizado público ou privado. Para ter informações sobre nomes de domínio personalizados para APIs privadas, consulte [the section called “Nomes de domínio personalizados para APIs privadas”](#).

### Registrar um nome de domínio

É necessário ter um nome de domínio da Internet registrado para configurar nomes de domínio personalizados para as APIs. É possível registrar o domínio da internet usando o [Amazon Route 53](#) ou um registrador de domínios de terceiros da sua escolha. O nome de domínio personalizado pode



ser o nome de um subdomínio ou do domínio raiz (também conhecido como “ápex da zona”) de um domínio da internet registrado.

O nome de domínio deve seguir a especificação [RFC 1035](#) e pode ter no máximo 63 octetos por etiqueta e 255 octetos no total.

Especificar o certificado para o nome de domínio personalizado

Antes de configurar um nome de domínio personalizado para uma API, você deve ter um certificado SSL/TLS pronto no ACM. Se o ACM não estiver disponível na região da AWS onde você está criando o nome de domínio personalizado, será necessário importar um certificado para o API Gateway nessa região.

Para importar um certificado SSL/TLS, você deve fornecer o corpo do certificado SSL/TLS formatado em PEM, sua chave privada e a cadeia de certificado para o nome de domínio personalizado.

Cada certificado armazenado no ACM é identificado por seu ARN. Com certificados emitidos pelo ACM, não é necessário se preocupar em expor detalhes de certificados confidenciais, como a chave privada. Para usar um certificado gerenciado pela AWS para um nome de domínio, basta fazer referência ao seu ARN.

Se o seu aplicativo usa a fixação de certificados, às vezes chamada de fixação SSL, para fixar um certificado do ACM, talvez o aplicativo não consiga se conectar ao seu domínio após a AWS renovar o certificado. Para ter mais informações, consulte [Problemas de fixação do certificado](#) no Guia do usuário do AWS Certificate Manager.

## Nomes de domínio personalizados curinga

Com nomes de domínio personalizados curinga, você pode suportar um número quase infinito de nomes de domínio sem exceder a [cota padrão](#). Por exemplo, você pode dar a cada um de seus clientes seu próprio nome de domínio, *customername*.example.com.

Para criar um nome de domínio personalizado curinga, especifique um curinga (\*) como o primeiro subdomínio de um domínio personalizado que representa todos os subdomínios possíveis de um domínio raiz.

Por exemplo, o nome de domínio curinga personalizado \*.example.com resulta em subdomínios, como a.example.com, b.example.com e c.example.com. Quando você cria o nome de domínio personalizado curinga, todos os subdomínios são roteados pelo modo de roteamento do nome de domínio curinga. Para rotear subdomínios para APIs diferentes, você pode executar uma das seguintes ações:

- Use regras de roteamento para encaminhar solicitações recebidas em \*.example.com para diferentes APIs REST de destino usando o cabeçalho Host. Para obter mais informações, consulte [the section called “Exemplo 4: regras de roteamento para nomes de domínio curinga”](#).
- Crie um nome de domínio para qualquer subdomínio que você queira rotear para um endpoint diferente. Em uma única conta da AWS, você pode ter \*.example.com e a.example.com.

Você pode usar as variáveis de contexto `$context.domainName` e `$context.domainPrefix` para determinar o nome de domínio que um cliente usou para chamar sua API. Para saber mais sobre variáveis de contexto, consulte [Variáveis para transformações de dados para o API Gateway](#).

Para criar um nome de domínio personalizado curinga, é necessário fornecer um certificado emitido pelo ACM que foi validado usando o DNS ou o método de validação por e-mail.

#### Note

Não é possível criar um nome de domínio personalizado curinga se uma conta da AWS diferente tiver criado um nome de domínio personalizado que esteja em conflito com o nome de domínio personalizado curinga. Por exemplo, se a conta A tiver criado a .example.com, a conta B não poderá criar o nome de domínio personalizado curinga \*.example.com. Se a conta A e a conta B compartilham um proprietário, entre em contato com a [Central de Suporte da AWS](#) para solicitar uma exceção.

## Próximas etapas para nomes de domínio personalizados

Veja as próximas etapas para nomes de domínio personalizados.

### Próximas etapas

- Para saber como configurar o certificado SSL/TLS, consulte [the section called “Preparar certificados no AWS Certificate Manager”](#).
- Para saber como criar um nome de domínio regional personalizado, consulte [the section called “Configurar um nome de domínio regional personalizado”](#).
- Para saber como criar um nome de domínio personalizado otimizado para borda, consulte [the section called “Configurar um nome de domínio personalizado otimizado para borda no API Gateway”](#).

- Para saber como migrar entre nomes de domínio personalizados regionais e otimizados para borda, consulte [the section called “Migrar nomes de domínios personalizados”](#).
- Para saber como conectar estágios de API a um nome de domínio personalizado, consulte [the section called “Encaminhar o tráfego às APIs por meio do nome de domínio personalizado no API Gateway”](#).
- Para saber como escolher uma política de segurança para o nome de domínio personalizado, consulte [the section called “Escolher uma política de segurança”](#).
- Para saber como desativar o endpoint padrão para o nome de domínio personalizado, consulte [the section called “Desabilitar o endpoint padrão”](#).
- Para saber como usar as verificações de integridade do Route 53 para controlar o failover de DNS por uma API do API Gateway, consulte [the section called “failover de DNS”](#).

Se for a primeira vez que você cria um nome de domínio personalizado, recomendamos começar com [the section called “Preparar certificados no AWS Certificate Manager”](#), para especificar o certificado e, depois, [the section called “Configurar um nome de domínio regional personalizado”](#), para criar um nome de domínio regional personalizado.

## Preparar certificados no AWS Certificate Manager

Antes de configurar um nome de domínio personalizado para uma API, você deve ter um certificado SSL/TLS pronto no AWS Certificate Manager. Para saber mais, consulte o [Guia do usuário do AWS Certificate Manager](#).

### Considerações

Veja a seguir algumas considerações para o certificado SSL/TLS.

- Se você criar um nome de domínio personalizado otimizado para borda, o API Gateway utilizará o CloudFront no suporte a certificados para nomes de domínio personalizados. Como tal, os requisitos e as restrições de um certificado SSL/TLS de nome de domínio personalizado são determinados pelo [CloudFront](#). Por exemplo, o tamanho máximo da chave pública é 2048, e o tamanho da chave privada pode ser de 1024, 2048 e 4096. O tamanho da chave pública é determinado pela autoridade de certificação que você utiliza. Peça à sua autoridade de certificação que retorne chaves de um tamanho diferente do comprimento padrão. Para obter mais informações, consulte [Acesso seguro aos seus objetos](#) e [Criar URLs e cookies assinados](#).
- Ao criar um nome de domínio personalizado regional, o tamanho máximo da chave pública deve ser 2048.

- Para usar um certificado do ACM com um nome de domínio regional personalizado, é necessário solicitar ou importar o certificado na mesma região da API. O certificado deve cobrir o nome de domínio personalizado.
- Para usar um certificado do ACM com um nome de domínio personalizado otimizado para borda, é necessário solicitar ou importar o certificado na região Leste dos EUA (N. da Virgínia): us-east-1.
- É necessário ter um nome de domínio registrado, como *example.com*. Você pode usar o [Amazon Route 53](#) ou um registrador de domínios credenciado de terceiros. Para obter uma lista de registradores, consulte o [Diretório de registradores acreditados](#) no site da ICANN.

### Como criar ou importar um certificado SSL/TLS no ACM

Os procedimentos a seguir mostram como criar ou importar um certificado SSL/TLS para um nome de domínio.

To request a certificate provided by ACM for a domain name

1. Faça login no [console do AWS Certificate Manager](#).
2. Selecione Request a certificate.
3. Em Tipo de certificado, escolha Solicitar um certificado público.
4. Escolha Próximo.
5. Em Nome de domínio totalmente qualificado, insira um nome de domínio personalizado para a API, por exemplo `api.example.com`.
6. Opcionalmente, escolha Add another name to this certificate.
7. Em Método de validação, escolha um método para validar a propriedade do domínio.
8. Em Algoritmo de chave, escolha um algoritmo de criptografia.
9. Escolha Solicitar.
10. Para uma solicitação válida, um proprietário registrado do domínio da Internet deve concordar com a solicitação antes que o ACM emita o certificado. Se você usa o Route 53 para gerenciar registros de DNS públicos, pode atualizar os registros diretamente do console do ACM.

## To import into ACM a certificate for a domain name

1. Obtenha um certificado SSL/TLS codificado em PEM para seu nome de domínio personalizado de uma autoridade de certificação (CA). Consulte uma lista parcial dessas CAs em [Mozilla Included CA List](#).
  - a. Gere uma chave privada para o certificado e salve a saída em um arquivo usando o toolkit [OpenSSL](#) no site da OpenSSL:

```
openssl genrsa -out private-key-file 2048
```

- b. Gere uma solicitação de assinatura de certificado (CSR) com a chave privada gerada anteriormente, usando o OpenSSL:

```
openssl req -new -sha256 -key private-key-file -out CSR-file
```

- c. Envie a CSR para a autoridade de certificação e salve o certificado resultante.
  - d. Baixe a cadeia de certificados da autoridade de certificação.

### Note

Se você obtiver a chave privada de outra maneira e a chave estiver criptografada, poderá usar o seguinte comando para descriptografar a chave antes de enviá-la ao API Gateway para a configuração de um nome de domínio personalizado.

```
openssl pkcs8 -topk8 -inform pem -in MyEncryptedKey.pem -outform pem -nocrypt -out MyDecryptedKey.pem
```

2. Carregue o certificado para o AWS Certificate Manager:
  - a. Faça login no [console do AWS Certificate Manager](#).
  - b. Selecione Importar um certificado.
  - c. Em Corpo do certificado, insira o corpo do certificado de servidor no formato PEM da autoridade de certificação. Veja a seguir um exemplo abreviado desse tipo de certificado.

```
-----BEGIN CERTIFICATE-----
EXAMPLECA+KgAwIBAgIQJ1XxJ8P1++g0fQtj0IBoqDANBgkqhkiG9w0BAQUFADBDB
```

```
...
az8Cg1aicxLBQ7EaWIhhgEXAMPLE
-----END CERTIFICATE-----
```

- d. Em Chave privada do certificado, insira a chave privada do certificado no formato PEM. Veja a seguir um exemplo abreviado desse tipo de chave.

```
-----BEGIN RSA PRIVATE KEY-----
EXAMPLEBAAKCAQEA2Qb3LDHD7StY7Wj6U2/opV6Xu37qUCCkeDWhwpZMYJ9/nET0
...
1qGvJ3u04vdnzaYN5WoyN5LFckr1A71+CszD1CGSqBVDWEXAMPLE
-----END RSA PRIVATE KEY-----
```

- e. Em Cadeia de certificados, insira os certificados intermediários no formato PEM e, opcionalmente, o certificado raiz, um após o outro, sem linhas em branco. Se você incluir o certificado raiz, sua cadeia de certificados deverá começar com certificados intermediários e terminar com o certificado raiz. Use os certificados intermediários fornecidos pela sua autoridade de certificação. Não inclua intermediários que não estejam no caminho da cadeia de confiança. O seguinte mostra um exemplo abreviado.

```
-----BEGIN CERTIFICATE-----
EXAMPLECA4ugAwIBAgIQWrYdrB5NogYUx1U9Pamy3DANBgkqhkiG9w0BAQUFADCB
...
8/ifB1IK3se2e4/hEfceEjX/arxbx1BJCHBv1EPNnsdw8EXAMPLE
-----END CERTIFICATE-----
```

Aqui está outro exemplo.

```
-----BEGIN CERTIFICATE-----
Intermediate certificate 2
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
Intermediate certificate 1
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
Optional: Root certificate
-----END CERTIFICATE-----
```

- f. Selecione Próximo e, depois, Próximo.

Depois que o certificado for criado ou importado com êxito, anote o ARN desse certificado. Você precisa dele ao configurar o nome de domínio personalizado.

## Configurar um nome de domínio regional personalizado no API Gateway

Use um nome de domínio regional personalizado para criar um URL de base de API fácil de usar. Com um nome de domínio regional personalizado, é possível mapear os estágios da API REST e HTTP para o mesmo nome de domínio personalizado e usar a autenticação TLS mútua.

### Considerações

Veja as considerações sobre o nome de domínio regional personalizado.

- É necessário fornecer um certificado do ACM específico da região. Esse certificado deve estar na mesma região que a API. Para obter mais informações sobre a criação ou upload de um certificado de nome de domínio personalizado, consulte [the section called “Preparar certificados no AWS Certificate Manager”](#).
- Quando você cria um nome de domínio regional personalizado (ou migra um) com um certificado do ACM, o API Gateway cria um perfil vinculado ao serviço na sua conta. A função vinculada ao serviço é necessária para anexar seu certificado do ACM ao seu endpoint regional. A função é denominada `AWSServiceRoleForAPIGateway` e tem a política gerenciada `APIGatewayServiceRolePolicy` anexada a ela. Para obter mais informações sobre como usar a função vinculada ao serviço, consulte [Usar funções vinculadas ao serviço](#).
- Depois de criar o nome de domínio regional personalizado, você deve criar um registro de DNS para apontar o nome de domínio personalizado para o domínio regional. Isso permite que o tráfego vinculado ao nome de domínio personalizado seja roteado para o nome de host regional da API.

O registro de DNS pode ser o CNAME ou um registro de alias A. Se você usa o Route 53 como provedor de DNS, crie um registro de alias A. Se você usa um provedor de DNS de terceiros, use um registro CNAME. Se você usar um registro CNAME e criar um endpoint da VPC de interface do API Gateway com DNS privado habilitado para uma API privada, não poderá resolver o nome de domínio personalizado na VPC que hospeda a API privada.

### Criar um nome de domínio regional personalizado

O procedimento a seguir mostra como criar um nome de domínio regional personalizado. Depois de concluir esse procedimento, crie uma regra de roteamento para rotear estágios da API para o nome de domínio personalizado.

## Console de gerenciamento da AWS

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha Custom Domain Names (Nomes de domínios personalizados) no painel de navegação principal.
3. Escolha Create (Criar).
4. Em Domain name (Nome de domínio), insira um nome de domínio
5. Em Modo de roteamento, escolha Somente regras de roteamento.

Nesse modo de roteamento, você só pode enviar tráfego do seu nome de domínio personalizado às suas APIs usando regras de roteamento. Para obter mais informações, consulte [the section called “Encaminhar o tráfego às APIs por meio do nome de domínio personalizado no API Gateway”](#).

6. Em Versão mínima do TLS, selecione uma versão.
7. Em Configuração de endpoint, em Tipo de endpoint de API, selecione Regional.
8. Excluir um certificado do ACM O certificado deve estar na mesma região que a API.
9. Escolha Criar.

## AWS CLI

O comando [create-domain-name](#) indicado abaixo cria um nome de domínio 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
```

A saída será exibida da seguinte forma:

```
{
 "ApiMappingSelectionExpression": "$request.basepath",
 "DomainName": "regional.example.com",
 "DomainNameConfigurations": [
 {
 "ApiGatewayDomainName": "d-numh1z56v6.execute-api.us-
west-2.amazonaws.com",
```



```
 "CertificateArn": "arn:aws:acm:us-
west-2:123456789012:certificate/123456789012-1234-1234-1234-12345678",
 "DomainNameStatus": "AVAILABLE",
 "EndpointType": "REGIONAL",
 "HostedZoneId": "Z20JLYMU09EFXC",
 "SecurityPolicy": "TLS_1_2"
 }
 "RoutingMode": "ROUTING_RULE_ONLY"
]
}
```

O valor da propriedade `DomainNameConfigurations` retorna o nome de host da API regional. Você deve criar um registro DNS para apontar seu nome de domínio personalizado para esse nome de domínio regional. Isso permite que o tráfego vinculado ao nome de domínio personalizado seja roteado para o nome de host dessa API regional.

Criar uma regra de roteamento para o nome de domínio regional personalizado

Depois de criar um nome de domínio personalizado, configure como o tráfego é encaminhado do nome de domínio personalizado às APIs. Como o modo de roteamento é definido como `ROUTING_RULE_ONLY`, você usa regras de roteamento para encaminhar às suas APIs as solicitações recebidas em seu nome de domínio personalizado.

Neste exemplo, você cria uma regra abrangente que encaminha a um estágio da API todas as solicitações recebidas em seu nome de domínio personalizado. Você também pode configurar regras de roteamento com base em diferentes condições de cabeçalho e caminho. Para obter mais informações, consulte [the section called “Regras de roteamento”](#).

Console de gerenciamento da AWS

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha um nome de domínio personalizado.
3. Na guia Detalhes do roteamento, escolha Adicionar regra de roteamento.
4. Escolha Adicionar uma nova condição para adicionar uma nova condição.
5. Mantenha essa regra sem nenhuma condição. Isso direciona todas as solicitações para seu nome de domínio personalizado à API de destino e ao estágio de destino.
6. Em Ação, use o menu suspenso para selecionar a API e o estágio de destino.

7. Escolha Próximo.
8. No campo de prioridade, insira **100**.

O API Gateway avalia as regras em ordem de prioridade, do valor mais baixo para o valor mais alto. Como essa é uma regra abrangente, você usa uma prioridade alta para que o API Gateway primeiro faça a correspondência com qualquer regra adicional que você criar primeiro.

9. Selecione Criar regra de roteamento.

## AWS CLI

O comando `create-routing-rule` apresentado abaixo cria uma regra de roteamento abrangente:

```
aws apigatewayv2 create-routing-rule \
 --domain-name 'regional.example.com' \
 --priority 100 \
 --conditions \
 --actions ' [{
 "InvokeApi": {
 "ApiId": "a1b2c3",
 "Stage": "prod"
 }
 }]'
```

Você pode alterar o modo de roteamento e criar outras regras a qualquer momento. Para obter mais informações, consulte [the section called “Encaminhar o tráfego às APIs por meio do nome de domínio personalizado no API Gateway”](#).

Criar um registro DNS para o nome de domínio regional personalizado

Depois de criar o nome de domínio personalizado e mapeamentos de caminho base, crie um registro de DNS para apontar o nome de domínio personalizado para o nome de domínio regional recém-criado.

Console de gerenciamento da AWS

Para usar o Console de gerenciamento da AWS, siga a documentação do Route 53 sobre [como configurar o Route 53 para rotear o tráfego para o API Gateway](#).

## AWS CLI

Para configurar os registros DNS para mapear o nome de domínio regional personalizado para o nome de host do ID de zona hospedada fornecido, primeiro crie um arquivo JSON que contenha a configuração de um registro DNS para o nome de domínio regional.

O `setup-dns-record.json` a seguir mostra como criar um registro DNS A para mapear um nome de domínio regional personalizado (`regional.example.com`) ao nome de host regional (`d-numh1z56v6.execute-api.us-west-2.amazonaws.com`) provisionado como parte da criação do nome de domínio personalizado. As propriedades `DNSName` e `HostedZoneId` de `AliasTarget` podem ter os valores `regionalDomainName` e `regionalHostedZoneId` respectivamente do nome de domínio personalizado. Também é possível obter os IDs de zona hospedada regional do Route 53 em [Endpoints e cotas do 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": "Z20JLYMU09EFXC",
 "EvaluateTargetHealth": false
 }
 }
 }
]
}
```

O comando [change-resource-record-sets](#) indicado abaixo cria um registro de DNS para o nome de domínio personalizado regional:

```
aws route53 change-resource-record-sets \
 --hosted-zone-id Z20JLYMU09EFXC \
 --change-batch file:///path/to/your/setup-dns-record.json
```

Substitua `hosted-zone-id` pelo ID da zona hospedada do Route 53 do conjunto de registros DNS na conta. O valor do parâmetro `change-batch` aponta para um arquivo JSON (*`setup-dns-record.json`*) em uma pasta (*`path/to/your`*).

## Configurar um nome de domínio personalizado otimizado para borda no API Gateway

Quando você cria um nome de domínio personalizado para uma API otimizada para borda, o API Gateway configura uma distribuição do CloudFront e um registro DNS para mapear o nome de domínio da API para o nome de domínio da distribuição do CloudFront. Solicitações para a API são roteadas para o API Gateway por meio da distribuição mapeada do CloudFront. Esse mapeamento refere-se a solicitações de API vinculadas ao nome de domínio personalizado a ser roteado para o API Gateway por meio da distribuição mapeada do CloudFront.

### Considerações

Veja algumas considerações sobre o nome de domínio personalizado otimizado para borda:

- Para configurar um nome de domínio personalizado otimizado para bordas ou atualizar seu certificado, você deve ter uma permissão para atualizar as distribuições do CloudFront.

As seguintes permissões são necessárias para atualizar as distribuições do CloudFront:

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "AllowCloudFrontUpdateDistribution",
 "Effect": "Allow",
 "Action": [
 "cloudfront:updateDistribution"
],
 "Resource": [
 "*"
]
 }
]
}
```

- É necessário solicitar ou importar um certificado para o nome de domínio personalizado otimizado para borda na região Leste dos EUA (N. da Virgínia): us-east-1.
- A distribuição do CloudFront criada pelo API Gateway pertence a uma conta específica da região afiliada ao API Gateway. Ao rastrear operações para criar e atualizar essa distribuição do CloudFront no CloudTrail, você deve usar esse ID de conta do API Gateway. Para obter mais

informações, consulte [Registrar a criação do nome de domínio personalizado no CloudTrail em log](#).

- O API Gateway oferece suporte a nomes de domínio personalizados otimizados para bordas, otimizando a Indicação de nome de servidor (SNI) na distribuição do CloudFront. Consulte mais informações sobre como usar nomes de domínio personalizados em uma distribuição do CloudFront, incluindo o formato de certificado necessário e o tamanho máximo de uma chave de certificado, em [Usar nomes de domínio alternativos e HTTPS](#) no Guia do desenvolvedor do Amazon CloudFront.
- Um nome de domínio personalizado otimizado para borda leva cerca de 40 minutos para ficar pronto.
- Após criar o nome de domínio personalizado otimizado para borda, é necessário criar um registro DNS para mapear o nome de domínio personalizado para o nome da distribuição do CloudFront.

### Criar um nome de domínio personalizado otimizado para borda

O procedimento a seguir descreve como criar um nome de domínio personalizado otimizado para borda para uma API.

### Console de gerenciamento da AWS

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha Custom Domain Names (Nomes de domínios personalizados) no painel de navegação principal.
3. Escolha Adicionar nome do domínio.
4. Em Nome de domínio, insira um nome de domínio
5. Em Modo de roteamento, escolha API\_MAPPING\_ONLY.
6. Em Tipo de endpoint de API, escolha Otimizado para a borda.
7. Escolha uma versão mínima de TLS.
8. Excluir um certificado do ACM
9. Escolha Adicionar nome do domínio.

## REST API

1. Chame [domainname:create](#), especificando o nome de domínio personalizado e o ARN de um certificado armazenado no AWS Certificate Manager.

A chamada de API bem-sucedida retorna uma resposta 201 Created que contém o ARN do certificado e o nome da distribuição do CloudFront associada em sua carga útil.

2. Observe o nome de domínio da distribuição do CloudFront mostrado na saída. Você precisará disso na próxima etapa para definir o destino do alias do registro A do domínio personalizado no seu DNS.

Para exemplos de código dessa chamada de API REST, consulte [domainname:create](#).

Um nome de domínio personalizado otimizado para borda leva cerca de 40 minutos para ficar pronto, mas o console exibe imediatamente o nome de domínio da distribuição do CloudFront associado, no formato *distribution-id*.cloudfront.net, junto com o ARN do certificado. Enquanto isso, você pode criar um mapeamento de caminho base ou uma regra de roteamento e configurar o registro de DNS para associar o nome de domínio personalizado ao nome de domínio da distribuição do CloudFront correspondente.

Configurar o mapeamento de caminho base de uma API com um nome de domínio personalizado como seu nome de host

Pelo fato de definir o modo de roteamento como API\_MAPPING\_ONLY, você pode usar o mapeamento de caminho base para usar um único nome de domínio personalizado como o nome de host de várias APIs. Isso torna uma API acessível por meio da combinação do nome de domínio personalizado e do caminho base associado.

Por exemplo, se, no API Gateway, você criou uma API chamada PetStore e outra API chamada Dogs e configurou um nome de domínio personalizado de api.example.com, pode definir o URL da API PetStore como https://api.example.com.

Isso associa a API PetStore ao caminho base de uma string vazia. Se você definir o URL da API PetStore como https://api.example.com/PetStore, isso associará a API PetStore ao caminho base de PetStore. É possível atribuir um caminho base de MyDogList para a API Dogs. A URL de https://api.example.com/MyDogList é então a URL raiz da API Dogs.

Para configurar mapeamentos de API em vários níveis, você só pode usar um nome de domínio regional personalizado. Os nomes de domínio personalizados otimizados para borda não são compatíveis. Para obter mais informações, consulte [the section called “Mapeamentos de API”](#).

O procedimento a seguir configura mapeamentos de API para mapear caminhos de seu nome de domínio personalizado para seus estágios de API.

### Console de gerenciamento da AWS

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha Custom domain names (Nomes de domínio personalizados) no painel de navegação principal do console do API Gateway.
3. Escolha um nome de domínio personalizado.
4. Escolha Configure API mappings (Configurar mapeamentos de API).
5. Escolha Add new mapping (Adicionar novo mapeamento).
6. Especifique a API, o Stage (Estágio) e o Path (Caminho) (opcional) para o mapeamento.
7. Escolha Salvar.

### REST API

Chame [basepathmapping:create](#) em um nome de domínio personalizado, especificando `basePath`, `restApiId` e uma propriedade `stage` de implantação na carga útil da solicitação.

A chamada de API bem-sucedida retorna uma resposta 201 Created.

Para exemplos de código da chamada de API REST, consulte [basepathmapping:create](#).

Para ter maior flexibilidade em relação a como você direciona o tráfego para as APIs, você pode alterar o modo de roteamento para `ROUTING_RULE_ONLY` ou `ROUTING_RULE_THEN_API_MAPPING` e criar uma regra de roteamento. Para obter mais informações, consulte [the section called “Encaminhar o tráfego às APIs por meio do nome de domínio personalizado no API Gateway”](#).

Criar um registro DNS para o nome de domínio personalizado otimizado para borda

Depois de iniciar a criação do nome de domínio personalizado otimizado para borda, configure o alias do registro DNS.

Recomendamos que você use o Route 53 para criar um alias de registro A para o nome de domínio personalizado e especifique o nome de domínio da distribuição do CloudFront como o destino do

alias. Isso significa que o Route 53 pode rotear seu nome de domínio personalizado, mesmo que seja um apex de zona. Para obter mais informações, consulte [Escolher entre conjuntos de registros de recursos de alias e não alias](#) no Guia do desenvolvedor do Amazon Route 53.

Para obter instruções sobre o Amazon Route 53, consulte [Routing traffic to an Amazon API Gateway API by using your domain name](#) no Guia do desenvolvedor do Amazon Route 53.

Registrar a criação do nome de domínio personalizado no CloudTrail em log

Quando o CloudTrail está habilitado para registrar em log as chamadas do API Gateway feitas pela sua conta, o API Gateway registra as atualizações da distribuição do CloudFront associadas quando um nome de domínio personalizado é criado ou atualizado para uma API. Esses logs estão disponíveis em us-east-1. Como essas distribuições do CloudFront são de propriedade do API Gateway, cada uma dessas distribuições do CloudFront reportadas é identificada por um dos seguintes IDs de conta do API Gateway específicos da região, e não pelo ID da conta do proprietário da API.

| Região       | ID da conta  |
|--------------|--------------|
| us-east-1    | 392220576650 |
| us-east-2    | 718770453195 |
| us-west-1    | 968246515281 |
| us-west-2    | 109351309407 |
| ca-central-1 | 796887884028 |
| eu-west-1    | 631144002099 |
| eu-west-2    | 544388816663 |
| eu-west-3    | 061510835048 |
| eu-central-1 | 474240146802 |
| eu-central-2 | 166639821150 |
| eu-north-1   | 394634713161 |



| Região         | ID da conta  |
|----------------|--------------|
| eu-south-1     | 753362059629 |
| eu-south-2     | 359345898052 |
| ap-northeast-1 | 969236854626 |
| ap-northeast-2 | 020402002396 |
| ap-northeast-3 | 360671645888 |
| ap-southeast-1 | 195145609632 |
| ap-southeast-2 | 798376113853 |
| ap-southeast-3 | 652364314486 |
| ap-southeast-4 | 849137399833 |
| ap-south-1     | 507069717855 |
| ap-south-2     | 644042651268 |
| ap-east-1      | 174803364771 |
| sa-east-1      | 287228555773 |
| me-south-1     | 855739686837 |
| me-central-1   | 614065512851 |

## Alternar um certificado importado para o ACM

O ACM lida automaticamente com a renovação dos certificados que ele emite. Não é necessário alternar certificados emitidos pelo ACM para seus nomes de domínio personalizados. O CloudFront se encarrega disso em seu nome.

No entanto, se você importar um certificado para o ACM e usá-lo para um nome de domínio personalizado, deverá alternar o certificado antes que ele expire. Isso envolve importar um novo certificado de terceiro para o nome de domínio e revezar o certificado existente para o novo. Você

precisará repetir o processo quando o certificado recém-importado expirar. Como alternativa, você pode solicitar que o ACM emita um novo certificado para o nome de domínio e alterne esse certificado existente com o novo certificado emitido pelo ACM. Depois disso, é possível deixar o ACM e o CloudFront encarregados de lidar com a alternância de certificados para você automaticamente. Para criar ou importar um novo certificado do ACM, siga as etapas em [the section called “Como criar ou importar um certificado SSL/TLS no ACM”](#).

O procedimento a seguir descreve como alternar um certificado para um nome de domínio.

#### Note

Leva cerca de 40 minutos para alternar um certificado importado no ACM.

### Console de gerenciamento da AWS

1. Solicite ou importe um certificado no ACM.
2. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
3. Escolha Custom domain names (Nomes de domínio personalizados) no painel de navegação principal do console do API Gateway.
4. Escolha um nome de domínio personalizado otimizado para borda.
5. Em Configuração de endpoint, escolha Editar.
6. Em Certificado do ACM, selecione um certificado na lista suspensa.
7. Escolha Salvar alterações para começar a alternar o certificado para o nome de domínio personalizado.

### REST API

Chame a ação [domainname:update](#), especificando o ARN do novo certificado do ACM para o nome de domínio especificado.

### AWS CLI

O comando [update-domain-name](#) abaixo atualiza o certificado do ACM para um nome de domínio otimizado para borda.

```
aws apigateway update-domain-name \
 --domain-name edge.example.com \
 --certificate-arn arn:aws:acm:us-east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012
```

```
--patch-operations "op='replace',path='/certificateArn',value='arn:aws:acm:us-east-2:111122223333:certificate/CERTEXAMPLE123EXAMPLE'"
```

O comando [update-domain-name](#) atualiza o certificado do ACM para um nome de domínio 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'"
```

Chamar a API com nomes de domínio personalizado ao usar um mapeamento de caminho base

Chamar uma API com um nome de domínio personalizado é o mesmo que chamá-la com o nome de domínio padrão, desde que a URL correta seja utilizada.

Os exemplos a seguir comparam e contrastam um conjunto de URLs padrão e as URLs personalizadas correspondentes de duas APIs (udxjef e qf3duz) em uma região especificada (us-east-1) e de um determinado nome de domínio personalizado (api.example.com).

| ID de API | Estágio | URL padrão                                              | Caminho base | URL personalizado                 |
|-----------|---------|---------------------------------------------------------|--------------|-----------------------------------|
| 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                    | /bookstore   | https://api.example.com/bookstore |

| ID de API | Estágio | URL padrão                                             | Caminho base | URL personalizado                 |
|-----------|---------|--------------------------------------------------------|--------------|-----------------------------------|
|           |         | .amazonaws.com/dev                                     |              |                                   |
| qf3duz    | tst     | https://qf3duz.execute-api.us-east-1.amazonaws.com/tst | /bookstand   | https://api.example.com/bookstand |

Para ter maior flexibilidade em relação a como você direciona o tráfego para suas APIs, você pode criar uma regra de roteamento. Para obter mais informações, consulte [the section called “Encaminhar o tráfego às APIs por meio do nome de domínio personalizado no API Gateway”](#).

O API Gateway oferece suporte a nomes de domínio personalizados para uma API usando a [Indicação de nome de servidor \(SNI\)](#). Você pode invocar a API com um nome de domínio personalizado usando um navegador ou uma biblioteca de cliente que ofereça suporte a SNIs.

O API Gateway impõe o SNI na distribuição do CloudFront. Para obter informações sobre como o CloudFront usa nomes de domínio personalizados, consulte [SSL personalizado no Amazon CloudFront](#).

## Migrar um nome de domínio personalizado para um tipo de endpoint de API diferente no API Gateway

Você pode migrar seu nome de domínio personalizado entre endpoints regionais e otimizados para bordas. Não é possível migrar um nome de domínio personalizado público para um nome de domínio personalizado privado. Primeiro adicione o novo tipo de configuração de endpoint à lista `endpointConfiguration.types` existente para o nome de domínio personalizado. Em seguida, configure um registro DNS para apontar o nome de domínio personalizado para o endpoint recém-provisionado. Por fim, você remove o endpoint obsoleto do nome de domínio personalizado.

### Considerações

Veja as considerações sobre como migrar o domínio personalizado entre um endpoint de API regional e um endpoint de API otimizado para borda:

- Um nome de domínio personalizado otimizado para borda exige um certificado fornecido pelo ACM da região Leste dos EUA (N. da Virgínia): us-east-1. Esse certificado é distribuído por todas as localizações geográficas.
- Um nome de domínio regional personalizado exige um certificado fornecido pelo ACM na mesma região que hospeda a API. Você pode migrar um nome de domínio personalizado otimizado para borda que não esteja na região us-east-1 para um nome de domínio regional personalizado solicitando um novo certificado do ACM da região da API.
- Pode levar até 60 segundos para concluir uma migração entre um nome de domínio personalizado otimizado para borda e um nome de domínio regional personalizado. O tempo de migração também depende de quando você atualiza os registros DNS.
- Você só poderá adicionar uma configuração do endpoint adicional se o modo de acesso ao endpoint estiver definido como BASIC. Assim que você tiver duas configurações do endpoint, não poderá mais alterar o modo de acesso ao endpoint. Para obter mais informações, consulte [the section called “Modo de acesso ao endpoint”](#).
- Se seu nome de domínio personalizado utiliza uma política de segurança que comece com SecurityPolicy\_, quando você adiciona um novo tipo de configuração do endpoint, o modo de acesso ao endpoint é o mesmo nos dois tipos, e você precisa escolher uma política de segurança que comece com SecurityPolicy\_ para o novo tipo de configuração do endpoint.

## Migrar nomes de domínios personalizados

### Note

Para concluir a migração, remova o endpoint obsoleto do nome de domínio personalizado.

O procedimento a seguir mostra como migrar um nome de domínio personalizado otimizado para borda para um nome de domínio regional personalizado.

## Console de gerenciamento da AWS

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha Custom Domain Names (Nomes de domínios personalizados) no painel de navegação principal.
3. Escolha um nome de domínio personalizado otimizado para borda.

4. Em Configuração de endpoint, escolha Editar.
5. Escolha Adicionar endpoint regional.
6. Em Certificado do ACM, escolha um certificado.

O certificado regional deve estar da mesma região da API regional.

7. Escolha Salvar alterações.
8. Configure um registro DNS para apontar o nome de domínio regional personalizado para esse nome de host regional. Consulte mais informações sobre [como configurar o Route 53 para encaminhar o tráfego ao API Gateway](#).
9. Depois de confirmar que a configuração de DNS está usando o endpoint correto, você exclui a configuração de endpoint otimizado para borda. Escolha o nome de domínio personalizado e, em Configuração de endpoint otimizado para borda, escolha Excluir.
10. Confirme sua escolha e exclua o endpoint.

## AWS CLI

O comando [update-domain-name](#) indicado abaixo migra o nome de domínio personalizado otimizado para borda para um nome de domínio 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" }
]'
```

O certificado regional deve ser da mesma região da API regional.

A saída será exibida da seguinte forma:

```
{
 "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"
 }
```

Para o nome de domínio personalizado regional migrado, a propriedade `regionalDomainName` resultante retorna o nome de host da API regional. Você deve configurar um registro DNS para apontar o nome de domínio personalizado regional para esse nome de host regional. Isso permite que o tráfego direcionado para o nome de domínio personalizado seja roteado para o host regional.

Depois que o registro DNS for definido, você poderá remover o nome de domínio personalizado otimizado para borda. O comando [update-domain-name](#) indicado abaixo remove o nome de domínio personalizado otimizado para borda.

```
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"}
]'
```

O procedimento a seguir mostra como migrar um nome de domínio personalizado otimizado para borda que use uma política de segurança aprimorada para um nome de domínio regional personalizado que também usa uma política de segurança aprimorada.

### Console de gerenciamento da AWS

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha Custom Domain Names (Nomes de domínios personalizados) no painel de navegação principal.
3. Escolha um nome de domínio personalizado otimizado para borda.

4. Em Configuração de endpoint, escolha Editar.
5. Escolha Adicionar endpoint regional.
6. Em Certificado do ACM, escolha um certificado.

O certificado regional deve estar da mesma região da API regional.

7. Em Política de segurança, escolha uma política de segurança que comece com SecurityPolicy\_.
8. Em Modo de acesso ao endpoint, escolha Básico.
9. Escolha Salvar alterações.
10. Configure um registro DNS para apontar o nome de domínio regional personalizado para esse nome de host regional. Consulte mais informações sobre [como configurar o Route 53 para encaminhar o tráfego ao API Gateway](#).
11. Depois de confirmar que a configuração de DNS está usando o endpoint correto, você exclui a configuração de endpoint otimizado para borda. Escolha o nome de domínio personalizado e, em Configuração de endpoint otimizado para borda, escolha Excluir.
12. Confirme sua escolha e exclua o endpoint.

## AWS CLI

O comando [update-domain-name](#) indicado abaixo migra o nome de domínio personalizado otimizado para borda para um nome de domínio 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" }
]'
```

O certificado regional deve ser da mesma região da API regional.

A saída será exibida da seguinte forma:

```
{
```



```

 "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"
 }
}

```

Para o nome de domínio personalizado regional migrado, a propriedade `regionalDomainName` resultante retorna o nome de host da API regional. Você deve configurar um registro DNS para apontar o nome de domínio personalizado regional para esse nome de host regional. Isso permite que o tráfego direcionado para o nome de domínio personalizado seja roteado para o host regional.

Depois que o registro DNS for definido, você poderá remover o nome de domínio personalizado otimizado para borda. O comando [update-domain-name](#) indicado abaixo remove o nome de domínio personalizado otimizado para borda.

```

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"}
]'

```

O procedimento a seguir mostra como migrar um nome de domínio personalizado regional para um nome de domínio personalizado otimizado para borda.

## Console de gerenciamento da AWS

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, selecione Nomes de domínio personalizados.
3. Escolha um nome de domínio regional personalizado.
4. Em Configuração de endpoint, escolha Editar.
5. Escolha Adicionar endpoint otimizado para borda.
6. Em Certificado do ACM, escolha um certificado.

O certificado de domínio otimizado para bordas deve ser criado na região us-east-1.

7. Selecione Salvar.
8. Configure um registro DNS para apontar o nome de domínio personalizado otimizado para borda para esse nome de host otimizado para borda. Consulte mais informações sobre [como configurar o Route 53 para encaminhar o tráfego ao API Gateway](#).
9. Depois de confirmar que a configuração de DNS está usando o endpoint correto, você exclui a configuração de endpoint regional. Escolha o nome de domínio personalizado e, em Configuração de endpoint (regional), escolha Excluir.
10. Confirme sua escolha e exclua o endpoint.

## AWS CLI

O comando [update-domain-name](#) indicado abaixo migra o nome de domínio personalizado regional para um nome de domínio personalizado otimizado para borda:

```
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" }
]'
```

O certificado de domínio otimizado para bordas deve ser criado na região us-east-1.

A saída será exibida da seguinte forma:

```
{
```

```

 "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 o nome de domínio personalizado especificado, o API Gateway retorna o nome de host da API otimizada para bordas como o valor da propriedade `distributionDomainName`. Você deve definir um registro DNS para apontar o nome de domínio personalizado otimizado para fronteiras para esse nome de domínio de distribuição. Isso permite que o tráfego direcionado para o nome de domínio personalizado otimizado para bordas seja roteado para o nome de host da API otimizada para bordas.

Depois que o registro DNS for definido, você poderá remover o tipo de endpoint `REGION` do nome de domínio personalizado. O comando [update-domain-name](#) indicado abaixo remove o tipo de endpoint regional:

```

aws apigateway update-domain-name \
 --domain-name api.example.com \
 --patch-operations '[
 {"op": "remove", "path": "/endpointConfiguration/types", "value": "REGIONAL"},
 {"op": "remove", "path": "regionalCertificateArn"}
]'

```

A saída será exibida como a seguir:

```

{
 "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"
}
}
```

## Encaminhar o tráfego às APIs por meio do nome de domínio personalizado no API Gateway

Ao configurar o modo de roteamento para o nome de domínio personalizado, você define como o tráfego de entrada é direcionado às APIs. Você envia tráfego às APIs usando regras de roteamento, mapeamentos de API ou regras de roteamento e mapeamentos de API. A seção a seguir explica quando usar regras de roteamento, quando usar mapeamentos de API e como definir o modo de roteamento para o nome de domínio personalizado.

### Quando usar regras de roteamento

Ao usar regras de roteamento, você direciona as solicitações recebidas que correspondem a determinadas condições para estágios específicos das APIs REST. Por exemplo, uma regra pode encaminhar uma solicitação ao estágio `production` da API REST `users` se ela contiver o cabeçalho `version:v1` e o caminho base `/users`. Use regras de roteamento para criar topologias avançadas de roteamento dinâmico que atendam a determinados casos de uso, como testes A/B ou aumento do uso de novas versões das APIs.

Recomendamos que, ao direcionar o tráfego para uma API REST, você use regras de roteamento para o nome de domínio personalizado. Você pode recriar qualquer mapeamento de API usando regras de roteamento. Para obter mais informações, consulte [the section called “Recriar um mapeamento de API usando regras de roteamento”](#).

Para APIs REST, você também pode usar regras de roteamento e mapeamentos de API ao mesmo tempo. Quando você usa regras de roteamento e mapeamentos de API ao mesmo tempo, o API Gateway sempre avalia as regras de roteamento antes de avaliar qualquer mapeamento de API. Use regras de roteamento e mapeamentos de API juntos para migrar seus nomes de domínio personalizados atuais ou para examinar as regras de roteamento.

### Considerações sobre regras de roteamento

As seguintes considerações podem afetar o uso de APIs privadas:

- As APIs HTTP ou de WebSocket não são permitidas como APIs de destino para regras de roteamento.
- Se o nome de domínio personalizado tiver mapeamentos de API para APIs REST e HTTP, não será possível usar regras de roteamento.
- Você pode criar uma regra de roteamento de um domínio personalizado privado para uma API REST privada. Você pode criar uma regra de roteamento para um domínio público personalizado para uma API regional ou otimizada para borda.
- Não é possível criar uma regra de roteamento de um domínio personalizado privado para uma API REST privada. Não é possível criar uma regra de roteamento de um domínio personalizado privado para uma API REST privada.

## Escolha entre regras de roteamento e mapeamentos de API

Recomendamos que, quando possível, você use regras de roteamento. Use mapeamentos de API somente para enviar tráfego a uma API HTTP ou de WebSocket.

## Definir o modo de roteamento para o nome de domínio personalizado

Você pode escolher qual modo de roteamento o API Gateway deve usar para rotear o tráfego para as APIs. Para obter mais informações, consulte [the section called “Encaminhar o tráfego às APIs por meio do nome de domínio personalizado no API Gateway”](#). Esta seção aborda modos de roteamento para nomes de domínio personalizados. Você deve definir um modo de roteamento para o nome de domínio personalizado para rotear o tráfego para as APIs. Os seguintes modos de roteamento são permitidos:

- **ROUTING\_RULE\_THEN\_API\_MAPPING**: use esse modo para enviar tráfego às APIs com regras de roteamento e mapeamentos de API. Nesse modo, todas as regras de roteamento têm prioridade sobre qualquer mapeamento de API. Para obter um exemplo desse modo, consulte [the section called “Exemplo 2: regras de roteamento e mapeamentos de API”](#).
- **ROUTING\_RULE\_ONLY**: use esse modo para permitir que somente as regras de roteamento enviem tráfego às APIs. Quando o nome de domínio personalizado usa esse modo, não é possível criar um mapeamento de API, mas você pode usar o comando [get-api-mappings](#) para visualizá-lo. Os autores de chamada de API não podem usar mapeamentos de API para acessar esse nome de domínio.
- **API\_MAPPING\_ONLY**: use esse modo para permitir que somente mapeamentos de API enviem tráfego às APIs. Quando o nome de domínio personalizado usa esse modo, não é possível criar

um mapeamento de API, mas você pode usar o comando `list-routing-rules` para visualizá-lo. Os autores de chamada de API não podem usar regras de roteamento para acessar esse nome de domínio.

Esse é o modo de roteamento padrão para todos os nomes de domínio existentes e quaisquer novos nomes de domínio que você criar.

Ao criar um nome de domínio personalizado usando `apigateway`, `API_MAPPING_ONLY` é chamado `BASE_PATH_MAPPING_ONLY` e `ROUTING_RULE_THEN_API_MAPPING` é chamado `ROUTING_RULE_THEN_BASE_PATH_MAPPING`. Esse comportamento ocorre somente na AWS CLI, no CloudFormation ou em qualquer SDK, não no Console de gerenciamento da AWS.

O procedimento a seguir mostra como alterar o modo de roteamento para um nome de domínio personalizado existente. Ao alterar o modo de roteamento do nome de domínio personalizado, os autores de chamada de API não podem acessar seu nome de domínio usando nenhum modo de roteamento incompatível.

#### Console de gerenciamento da AWS

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha Custom Domain Names (Nomes de domínios personalizados) no painel de navegação principal.
3. Escolha um nome de domínio personalizado.
4. Em Detalhes do domínio, escolha Editar.
5. Em Modo de roteamento, escolha `ROUTING_RULE_THEN_API_MAPPING`.
6. Escolha Salvar.

Se você alterar o modo de roteamento para `ROUTING_RULE_ONLY` ou `API_MAPPING_ONLY`, todos os mapeamentos de API ou regras de roteamento que você criou serão removidos da página de detalhes do nome de domínio do console. Se você alterar o modo de roteamento para permitir o uso de regras de roteamento ou mapeamentos de API, esses recursos retornarão.

#### AWS CLI - `apigatewayv2`

O comando [update-domain-name](#) indicado abaixo atualiza um nome de domínio para usar o modo de roteamento `ROUTING_RULE_THEN_API_MAPPING`:

```
aws apigatewayv2 update-domain-name \
 --domain-name 'api.example.com' \
 --routing-mode "ROUTING_RULE_THEN_API_MAPPING"
```

A saída será exibida da seguinte forma:

```
{
 "ApiMappingSelectionExpression": "$request.basepath",
 "DomainName": "api.example.com",
 "DomainNameArn": "arn:aws:apigateway:us-west-2::/domainnames/api.example.com",
 "DomainNameConfigurations": [
 {
 "ApiGatewayDomainName": "d-abcdefg.execute-api.us-west-2.amazonaws.com",
 "CertificateArn": "arn:aws:acm:us-west-2:111122223333:certificate/
abcdefg-123456-abcdefg",
 "DomainNameStatus": "AVAILABLE",
 "EndpointType": "REGIONAL",
 "HostedZoneId": "Z20JLYMU09EFXC",
 "SecurityPolicy": "TLS_1_2"
 }
],
 "RoutingMode": "ROUTING_RULE_THEN_API_MAPPING",
 "Tags": {}
}
```

## AWS CLI - apigateway

O comando [update-domain-name](#) indicado abaixo atualiza um nome de domínio personalizado privado para usar o modo de roteamento 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'"
```

A saída será exibida da seguinte forma:

```
{
 "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"
}
```

## Regras de roteamento para associar estágios da API a um nome de domínio personalizado para APIs REST

Uma regra de roteamento é um conjunto de condições que, quando correspondidas, invocam uma ação. Por exemplo, uma regra pode rotear qualquer solicitação recebida em um nome de domínio personalizado que contenha o cabeçalho `Hello:World` e o caminho base `users` para o estágio `production` de uma API REST.

As regras são avaliadas em ordem de prioridade e, se você definir o modo de roteamento com o `ROUTING_RULE_THEN_API_MAPPING`, o API Gateway sempre avaliará todas as regras de roteamento antes de avaliar qualquer mapeamento de API. A lista a seguir descreve como uma regra de roteamento usa condições, ações e prioridades.

### Condições

Quando as condições de uma regra forem atendidas, a ação será executada. O API Gateway permite até duas condições de cabeçalho e uma condição de caminho. O API Gateway avalia as condições do cabeçalho e as condições do caminho base ao mesmo tempo.

Você pode criar uma regra sem nenhuma condição. Quando o API Gateway avalia essa regra, a ação sempre é executada. Você pode criar uma regra sem nenhuma condição como uma regra abrangente.

Para ter mais informações sobre condições de cabeçalho, consulte [the section called “Correspondência com condições de cabeçalho”](#). Para ter mais informações sobre as condições de caminho, consulte [the section called “Correspondência com condições de caminho”](#).



## Ações

As ações resultam da correspondência entre condições e uma regra de roteamento. No momento, a única ação possível é invocar um estágio de uma API REST.

Cada regra pode ter uma ação.

## Prioridade

A prioridade determina em qual ordem as regras são avaliadas, do valor mais baixo para o valor mais alto. As regras não podem ter a mesma prioridade.

Você pode definir a prioridade de 1 a 1.000.000. Se a prioridade de uma regra for 1, o API Gateway a avaliará primeiro. Recomendamos que, ao criar uma regra, você adicione intervalos nas prioridades. Isso ajuda você a mudar a prioridade das regras e adicionar novas regras. Para obter mais informações, consulte [the section called “Alterar a prioridade de uma regra de roteamento”](#).

Para ver exemplos de como o API Gateway avalia as regras de roteamento, consulte [the section called “Exemplos de como o API Gateway avalia regras de roteamento”](#).

## Tipos de condição da regra de roteamento do API Gateway

A seção a seguir descreve os tipos de condição de regra de roteamento. O API Gateway só encontrará correspondência com uma regra se todas as condições forem verdadeiras.

### Correspondência com condições de cabeçalho

Ao criar uma condição de cabeçalho, você pode fazer a correspondência entre o nome do cabeçalho e o valor glob do cabeçalho, como `Hello:World`. O API Gateway usa uma correspondência literal para validar a correspondência com condições de cabeçalho. Sua condição pode utilizar até dois cabeçalhos usando AND entre eles. Por exemplo, pode haver correspondência com sua condição se uma solicitação recebida contiver `Hello:World` e `x-version:beta`.

A correspondência do nome do cabeçalho não diferencia maiúsculas de minúsculas, mas o valor glob do cabeçalho diferencia maiúsculas de minúsculas. `Hello:World` corresponderá a `hello:World`, mas não a `Hello:world`.

Para obter uma lista de valores de cabeçalho restritos, consulte [the section called “Restrições”](#).

## Uso de curingas com condições de cabeçalho

Você só pode usar curingas no valor glob do cabeçalho, e eles devem ser `*prefix-match`, `suffix-match*` ou `*contains*`. A tabela a seguir mostra exemplos de como usar curingas para fazer a correspondência com as condições do cabeçalho.

| Condições de cabeçalho                                    | Solicitações que correspondem à regra de roteamento                                                                                                                                                            | Solicitações que não correspondem à regra de roteamento                                                                                                                                                         |
|-----------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x-version: a*</code>                                | <ul style="list-style-type: none"> <li>• <code>x-version: account</code></li> <li>• <code>x-version: alpha</code></li> </ul>                                                                                   | <ul style="list-style-type: none"> <li>• <code>x-version: backup</code></li> <li>• <code>x-version: beta</code></li> <li>• <code>x-version: users</code></li> </ul>                                             |
| <code>x-version: *a</code>                                | <ul style="list-style-type: none"> <li>• <code>x-version: alpha</code></li> <li>• <code>x-version: beta</code></li> </ul>                                                                                      | <ul style="list-style-type: none"> <li>• <code>x-version: account</code></li> <li>• <code>x-version: backup</code></li> <li>• <code>x-version: users</code></li> </ul>                                          |
| <code>x-version: *a*</code>                               | <ul style="list-style-type: none"> <li>• <code>x-version: account</code></li> <li>• <code>x-version: alpha</code></li> <li>• <code>x-version: backup</code></li> <li>• <code>x-version: beta</code></li> </ul> | <ul style="list-style-type: none"> <li>• <code>x-version: users</code></li> </ul>                                                                                                                               |
| <code>x-version: *a*</code> e <code>x-version: *b*</code> | <ul style="list-style-type: none"> <li>• <code>x-version: backup</code></li> <li>• <code>x-version: beta</code></li> </ul>                                                                                     | <ul style="list-style-type: none"> <li>• <code>x-version: account</code></li> <li>• <code>x-version: alpha</code></li> <li>• <code>x-version: users</code></li> </ul>                                           |
| <code>x-version: b*</code> e <code>x-version: *a</code>   | <ul style="list-style-type: none"> <li>• <code>x-version: beta</code></li> </ul>                                                                                                                               | <ul style="list-style-type: none"> <li>• <code>x-version: account</code></li> <li>• <code>x-version: alpha</code></li> <li>• <code>x-version: backup</code></li> <li>• <code>x-version: users</code></li> </ul> |
| <code>x-version: *</code>                                 | <ul style="list-style-type: none"> <li>• <code>x-version: account</code></li> <li>• <code>x-version: alpha</code></li> <li>• <code>x-version: backup</code></li> </ul>                                         | Nenhum                                                                                                                                                                                                          |

| Condições de cabeçalho | Solicitações que correspondem à regra de roteamento                                                                       | Solicitações que não correspondem à regra de roteamento |
|------------------------|---------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------|
|                        | <ul style="list-style-type: none"> <li>• <code>x-version: beta</code></li> <li>• <code>x-version: users</code></li> </ul> |                                                         |

Se você criar condições para vários valores de cabeçalho, como `Accept: application/json, text/xml`, é recomendável usar `*contains*` para as condições de cabeçalho e evitar criar condições usando o caractere vírgula (,).

Como o API Gateway faz a correspondência com as condições de cabeçalho de maneira literal, as correspondências semânticas podem ser roteadas de forma diferente. A tabela a seguir mostra a diferença nos resultados das regras de roteamento.

| Condições de cabeçalho      | Solicitações que correspondem à regra de roteamento                                                                                                                                           | Solicitações que não correspondem à regra de roteamento                                             |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------|
| <code>Accept: *json</code>  | <ul style="list-style-type: none"> <li>• <code>Accept: application/json</code></li> <li>• <code>Accept: text/xml</code></li> </ul>                                                            | <ul style="list-style-type: none"> <li>• <code>Accept: application/json, text/xml</code></li> </ul> |
| <code>Accept: *json*</code> | <ul style="list-style-type: none"> <li>• <code>Accept: application/json</code></li> <li>• <code>Accept: text/xml</code></li> <li>• <code>Accept: application/json, text/xml</code></li> </ul> | Nenhum                                                                                              |

## Correspondência com condições de caminho

Quando você cria uma condição de caminho base, se a solicitação de entrada contiver o caminho especificado, haverá correspondência com a regra. A correspondência diferencia maiúsculas de minúsculas, portanto, o caminho `New/Users` não corresponderá a `new/users`.

Você pode criar uma condição de caminho base somente para um caminho base.

Para obter uma lista de condições restritas de caminho base, consulte [the section called “Restrições”](#).

## Remoção do caminho base com as condições do caminho base

Ao criar uma condição de caminho base, você pode optar por remover o caminho base. Ao remover o caminho base, o API Gateway remove o caminho base correspondente de entrada ao invocar a API de destino. Esse comportamento é igual ao de quando você usa um mapeamento de API. Quando você não remove o caminho base, o API Gateway encaminha o caminho base completo para a API de destino. Recomendamos que você remova o caminho base somente ao recriar um mapeamento de API.

A tabela a seguir mostra exemplos de como o API Gateway avalia a condição de remoção do caminho base.

| Condição                                        | Remover o caminho base | Solicitação de entrada                       | Resultado                                                            |
|-------------------------------------------------|------------------------|----------------------------------------------|----------------------------------------------------------------------|
| Se o caminho base contiver PetStoreShopper/dogs | Verdadeiro             | GET https://example.com/PetStoreShopper/dogs | O API Gateway chamará o método GET do recurso /.                     |
| Se o caminho base contiver PetStoreShopper/dogs | Falso                  | GET https://example.com/PetStoreShopper/dogs | O API Gateway chamará o método GET do recurso PetStoreShopper/dogs . |
| Se o caminho base contiver PetStoreShopper      | Verdadeiro             | GET https://example.com/PetStoreShopper/dogs | O API Gateway chamará o método GET do recurso dogs.                  |
| Se o caminho base contiver PetStoreShopper      | Falso                  | GET https://example.com/PetStoreShopper/dogs | O API Gateway chamará o método GET do recurso PetStoreShopper/dogs . |

| Condição                                   | Remover o caminho base | Solicitação de entrada                                  | Resultado                                                                                                               |
|--------------------------------------------|------------------------|---------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|
| Se o caminho base contiver PetStoreShopper | Verdadeiro             | GET https://example.com/PetStoreShopper?birds=available | O API Gateway chamará o método GET do recurso / com o parâmetro de string de consulta birds=available .                 |
| Se o caminho base contiver PetStoreShopper | Falso                  | GET https://example.com/PetStoreShopper?birds=available | O API Gateway chamará o GET método do / PetStoreShopper recurso com o parâmetro de string de consulta birds=available . |

## Restrições

- A API de destino e o nome de domínio personalizado devem estar na mesma conta da AWS.
- Cada regra pode ter uma API de destino.
- Você só pode criar uma regra de roteamento entre um nome de domínio personalizado privado e uma API privada e entre um nome de domínio personalizado público e uma API pública. Não é permitido misturar recursos públicos e privados.
- Se o nome de domínio personalizado tiver mapeamentos de API para APIs REST e HTTP, não será possível usar regras de roteamento.
- O número máximo de prioridade é 1.000.000.
- Restrições de cabeçalho:
  - Cada condição anyOf só pode conter um valor de cabeçalho.
  - Os únicos caracteres permitidos para nomes de cabeçalho e valores glob de cabeçalho são especificados pela [RFC 7230](#), que são a-z, A-Z e 0-9, e estes caracteres especiais: \*? - ! # \$ % & ' . ^ \_ ` | ~.

- Você só pode usar um curinga no valor glob do cabeçalho, mas ele deve ser `*prefix-match`, `suffix-match*` ou `*contains*`. Não é possível usar `*` no meio de um valor glob de cabeçalho.
- Nomes de cabeçalho curinga não são permitidos.
- O nome do cabeçalho deve ter menos de 40 caracteres.
- O valor glob do cabeçalho deve ter menos de 128 caracteres.
- O valor glob do cabeçalho para uma correspondência infix deve ter menos de 40 caracteres.
- Os seguintes cabeçalhos não são permitidos como condições:
  - `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
- Restrições do caminho base:
  - O caminho base deve ter menos de 128 caracteres.
  - O caminho base deve conter apenas letras, números e os seguintes caracteres: `$-_.+!*'()/.` Esses caracteres não são compatíveis com expressões regulares (regex).
  - O caminho base não pode começar ou terminar com o caractere de barra invertida (`\`).

### Exemplos de como o API Gateway avalia regras de roteamento

A seção a seguir mostra quatro exemplos de como o API Gateway avalia regras de roteamento e mapeamentos de API.

#### Exemplo 1: somente regras de roteamento

Neste exemplo, o nome de domínio personalizado `https://petstore.example.com` tem o modo de roteamento definido como `ROUTING_RULE_ONLY` e as regras e prioridades de roteamento a seguir.

| ID da regra | Prioridade | Condições                                                                                                                           | Ação             |
|-------------|------------|-------------------------------------------------------------------------------------------------------------------------------------|------------------|
| abc123      | 10         | Se a solicitação<br>contiver cabeçalho:<br>Hello:World                                                                              | API de destino 1 |
| zzz000      | 50         | Se a solicitação<br>contiver cabeçalhos<br>Accept:image/<br>webp e Pet:Dog-*<br>e se o caminho base<br>contiver PetStoreS<br>hopper | API de destino 2 |
| efg456      | 100        | Nenhum                                                                                                                              | API de destino 3 |

A tabela a seguir mostra como o API Gateway aplica as regras de roteamento anteriores a solicitações de exemplo.

| Solicitação                                                                                                      | API selecionada  | Explicação                                                                                                                                                                                                                                                                                                                                                                                                            |
|------------------------------------------------------------------------------------------------------------------|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>https://petstore.example.com -h "Hello:World"</code>                                                       | API de destino 1 | As solicitações correspondem à regra de roteamento abc123.                                                                                                                                                                                                                                                                                                                                                            |
| <code>https://petstore.example.com/PetStoreShopper -h "Hello:World", "Pet:Dog-Bella", "Accept:image/webp"</code> | API de destino 1 | <p>O API Gateway avalia todas as regras de roteamento em ordem de prioridade. A regra de roteamento abc123 é a primeira em prioridade e as condições correspondem, então o API Gateway invoca a API de destino 1.</p> <p>Embora as condições da solicitação também correspondam à regra de roteamento zzz000, o API Gateway não avalia nenhuma outra regra de roteamento depois que encontra uma correspondência.</p> |
| <code>https://petstore.example.com/PetStoreShopper -h "Pet:Dog-Bella", "Accept:image/webp"</code>                | API de destino 2 | As solicitações correspondem à regra de roteamento zzz000. Houve correspondência porque Pet:Dog-Bella era uma string compatível com Pet:Dog-*                                                                                                                                                                                                                                                                         |
| <code>https://petstore.example.com/PetStoreShopper -h "Pet:Dog-Bella"</code>                                     | API de destino 3 | A solicitação não corresponde à regra de roteamento abc123. A solicitação não corresponde à regra de                                                                                                                                                                                                                                                                                                                  |



| Solicitação | API selecionada | Explicação                                                                                                                                                                                              |
|-------------|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|             |                 | roteamento zzz000 porque todos os cabeçalhos necessários não estão presentes. A próxima regra prioritária corresponde a todas as solicitações recebidas, então o API Gateway invoca a API de destino 3. |

## Exemplo 2: regras de roteamento e mapeamentos de API

Neste exemplo, o nome de domínio personalizado `https://petstore.diagram.example.com` tem o modo de roteamento definido como `ROUTING_RULE_THEN_API_MAPPING` e as regras de roteamento e mapeamentos de API a seguir.

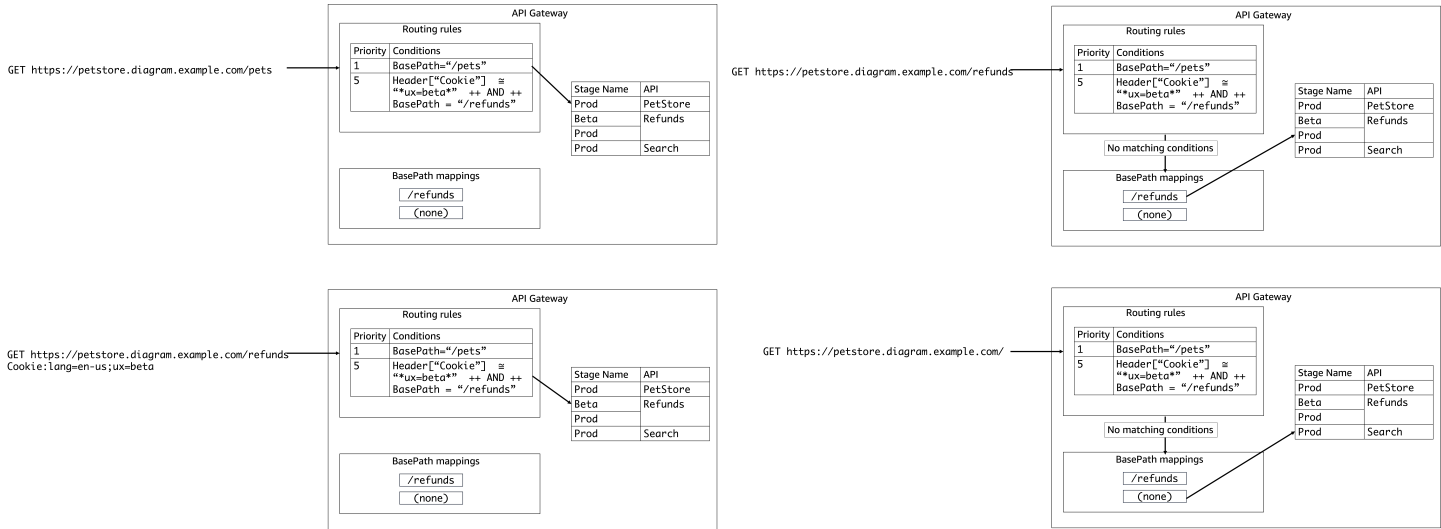
| ID da regra | Prioridade | Condições                                                                                                                            | Ação                                    |
|-------------|------------|--------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------|
| abc123      | 1          | Se a solicitação contiver <code>pets</code>                                                                                          | Invoque o estágio Prod da API PetStore. |
| 000zzz      | 5          | Se a solicitação contiver cabeçalhos <code>Cookie</code> e <code>*ux=beta*</code> e se o caminho base contiver <code>/refunds</code> | Invoque o estágio Beta da API Refunds.  |

A tabela a seguir mostra mapeamentos de API para `https://petstore.backup.example.com`.

| Mapeamento de API     | API selecionada                        |
|-----------------------|----------------------------------------|
| <code>/refunds</code> | Invoque o estágio Prod da API Refunds. |

| Mapeamento de API | API selecionada                       |
|-------------------|---------------------------------------|
| (none)            | Invoque o estágio Prod da API Search. |

O diagrama a seguir mostra como o API Gateway aplica as regras de roteamento e os mapeamentos de API anteriores a solicitações de exemplo. As solicitações de exemplo estão resumidas na tabela após esse diagrama.



A tabela a seguir mostra como o API Gateway aplica as regras de roteamento e os mapeamentos de API anteriores a solicitações de exemplo.

| Solicitação                                                                               | API selecionada                 | Explicação                                                                                                                                                                                  |
|-------------------------------------------------------------------------------------------|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>https://petstore.diagram.com/pets</code>                                            | O estágio Prod da API PetStore. | A solicitação corresponde à regra de roteamento abc123.                                                                                                                                     |
| <code>https://petstore.diagram.example.com/refunds -h "Cookie: lang=en-us;ux=beta"</code> | O estágio Beta da API Refunds.  | A solicitação corresponde à regra de roteamento 000zzz. O cabeçalho Cookie contém a correspondência <code>*contains*</code> correta e a correspondência do caminho base para essa condição. |

| Solicitação                                                         | API selecionada                | Explicação                                                                                                                                                                                                                                                                                                                        |
|---------------------------------------------------------------------|--------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>https://petstore.d<br/>iagram.example.com/<br/>refunds</code> | O estágio Prod da API Refunds. | A solicitação não tem os cabeçalhos necessários para fazer a correspondência com a regra de roteamento <code>zzz000</code> . Se o API Gateway não conseguir fazer a correspondência com uma regra de roteamento, ele retornará aos mapeamentos de API. O API Gateway pode associar o caminho base ao estágio Prod da API Refunds. |
| <code>https://petstore.d<br/>iagram.example.com/</code>             | O estágio Prod da API Search.  | A solicitação faz a correspondência entre o mapeamento de API e o caminho vazio (none).                                                                                                                                                                                                                                           |

### Exemplo 3: regras de roteamento e mapeamentos de API com vários níveis

Neste exemplo, o nome de domínio personalizado `https://petstore.backup.example.com` tem o modo de roteamento definido como `ROUTING_RULE_THEN_API_MAPPING` e as regras de roteamento e mapeamentos de API a seguir.

A tabela a seguir mostra regras de roteamento para `https://petstore.backup.example.com`.

| ID da regra         | Prioridade | Condições                                                     | Ação             |
|---------------------|------------|---------------------------------------------------------------|------------------|
| <code>abc123</code> | 10         | Se a solicitação contiver cabeçalho: <code>Hello:World</code> | API de destino 1 |
| <code>000zzz</code> | 50         | Se a solicitação contiver cabeçalhos                          | API de destino 2 |

| ID da regra | Prioridade | Condições                                                                  | Ação |
|-------------|------------|----------------------------------------------------------------------------|------|
|             |            | Accept:image/webp e Pet:Dog-* e se o caminho base contiver PetStoreShopper |      |

A tabela a seguir mostra mapeamentos de API para `https://petstore.backup.example.com`.

| Mapeamento de API    | API selecionada  |
|----------------------|------------------|
| PetStoreShopper      | API de destino 3 |
| PetStoreShopper/cats | API de destino 4 |

A tabela a seguir mostra como o API Gateway aplica as regras de roteamento e os mapeamentos de API anteriores a solicitações de exemplo.

| Solicitação                                                                                  | API selecionada  | Explicação                                                                                                                                                                                                                                                                                                            |
|----------------------------------------------------------------------------------------------|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>https://petstore.example.com/PetStoreShopper -h "Accept:image/webp", "Pet:Cats"</code> | API de destino 3 | A solicitação não tem os cabeçalhos necessários para fazer a correspondência com a regra de roteamento <code>zzz000</code> . Se o API Gateway não conseguir fazer a correspondência com uma regra de roteamento, ele retornará aos mapeamentos de API. O API Gateway pode associar o caminho base à API de destino 3. |
| <code>https://petstore.example.com/PetStore</code>                                           | API de destino 1 | A solicitação corresponde à regra de roteamento <code>abc123</code> .                                                                                                                                                                                                                                                 |

| Solicitação                                              | API selecionada | Explicação                                                                                                                                                                                                                                |
|----------------------------------------------------------|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| eShopper/cats -h<br>"Hello:World"                        |                 | Se o modo de roteamento estiver definido como <code>ROUTING_RULE_THEN_API_MAPPING</code> , as regras de roteamento sempre terão prioridade sobre os mapeamentos de API.                                                                   |
| https://petstore.example.com/Admin -h<br>"Pet:Dog-Bella" | Nenhum          | A solicitação não corresponde a nenhuma regra de roteamento ou mapeamento de API. Como não há uma regra de roteamento padrão, o API Gateway rejeita a chamada e envia um código de status 403 <code>Forbidden</code> ao autor da chamada. |

#### Exemplo 4: regras de roteamento para nomes de domínio curinga

Neste exemplo, o nome de domínio personalizado `https://*.example.com` é um nome de domínio curinga. O curinga aceita todos os subdomínios que são roteados de volta para o mesmo domínio. Os exemplos de regra de roteamento a seguir alteram esse comportamento para permitir que subdomínios sejam roteados para diferentes APIs de destino usando o cabeçalho `Host`.

A tabela a seguir mostra regras de roteamento para `https://*.example.com`.

| ID da regra | Prioridade | Condições                                                            | Ação             |
|-------------|------------|----------------------------------------------------------------------|------------------|
| abc123      | 10         | Se a solicitação contiver cabeçalho <code>:Host:a.example.com</code> | API de destino 1 |
| 000zzz      | 50         | Se a solicitação contiver cabeçalho                                  | API de destino 2 |

| ID da regra | Prioridade | Condições              | Ação             |
|-------------|------------|------------------------|------------------|
|             |            | : Host : b.example.com |                  |
| efg456      | 500        | Nenhum                 | API de destino 3 |

A tabela a seguir mostra como o API Gateway aplica as regras de roteamento anteriores a solicitações de exemplo.

| Solicitação                 | API selecionada  | Explicação                                                                                    |
|-----------------------------|------------------|-----------------------------------------------------------------------------------------------|
| https://a.example.com       | API de destino 1 | O cabeçalho Host é a.example.com . Essa solicitação corresponde à regra de roteamento abc123. |
| https://b.example.com       | API de destino 2 | O cabeçalho Host é b.example.com . Essa solicitação corresponde à regra de roteamento 000zzz. |
| https://testing.example.com | API de destino 3 | Isso corresponde à regra de roteamento abrangente efg456.                                     |

## Quando usar regras de roteamento

Você pode criar uma regra de roteamento usando o Console de gerenciamento da AWS, a AWS CLI ou qualquer SDK da AWS. Depois que você criar um trabalho, não poderá mais alterar ou atualizar a respectiva prioridade.

### Criar uma regra de roteamento

O procedimento a seguir mostra como criar uma regra de roteamento para um nome de domínio personalizado com um modo de roteamento definido como `ROUTING_RULE_THEN_API_MAPPING` ou `ROUTING_RULE_ONLY`.

## Console de gerenciamento da AWS

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha Custom Domain Names (Nomes de domínios personalizados) no painel de navegação principal.
3. Escolha um nome de domínio personalizado.
4. Na guia Detalhes do roteamento, escolha Adicionar regra de roteamento.
5. Escolha Adicionar uma nova condição para adicionar uma nova condição.

Você pode adicionar uma condição de cabeçalho ou de caminho base. Para fazer a correspondência de todas as solicitações recebidas com seu nome de domínio personalizado, não adicione uma condição.

6. Em Ação, use o menu suspenso para selecionar a API e o estágio de destino.
7. Escolha Próximo.
8. No campo de prioridade, insira um número para sua prioridade.

O API Gateway avalia as regras em ordem de prioridade, do valor mais baixo para o valor mais alto.

Se você estiver criando uma regra sem uma condição, recomendamos usar uma prioridade com um valor alto.

9. Selecione Criar regra de roteamento.

## AWS CLI

O comando [create-routing-rule](#) indicado abaixo cria uma regra de roteamento com prioridade 50. Neste exemplo, o API Gateway roteia todas as solicitações recebidas que tenham os cabeçalhos `Hello:World` e `x-version:beta` e o caminho base `PetStoreShopper` para a 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"
 }
}
]'
```

A saída será exibida da seguinte forma:

```

{
 "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"
}

```

## Alterar a prioridade de uma regra de roteamento

Você pode alterar a prioridade de uma regra de roteamento. Essa alteração entra em vigor imediatamente e pode afetar a forma como os consumidores da API invocam seus nomes de domínio personalizados. Recomendamos que, ao definir a prioridade de suas regras de roteamento, você deixe intervalos entre as regras.

Por exemplo, considere duas regras de roteamento, regra abc123 com prioridade 50 e regra zzz000 com prioridade 150. Para alterar a prioridade das regras de modo que o API Gateway avalie a regra zzz000 primeiro, você pode alterar a prioridade da regra zzz000 para 30.

### Console de gerenciamento da AWS

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha Custom Domain Names (Nomes de domínios personalizados) no painel de navegação principal.
3. Escolha um nome de domínio personalizado.
4. Na guia Detalhes do roteamento, escolha sua regra de roteamento e selecione Editar.
5. Escolha Próximo.
6. Em prioridade, insira a nova prioridade.
7. Escolha Salvar alterações.

### AWS CLI

O comando [put-routing-rule](#) indicado abaixo altera a prioridade de uma regra de roteamento 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"
 }
}
]'
```

A saída será exibida da seguinte forma:

```

{
 "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"
}
```

## Recriar um mapeamento de API usando regras de roteamento

Você pode recriar um mapeamento de API usando regras de roteamento. Para recriar um mapeamento de API, ative a remoção do caminho base. Isso preserva o comportamento dos mapeamentos de API. Para obter mais informações, consulte [the section called “Remoção do caminho base com as condições do caminho base”](#).

O tutorial a seguir mostra como recriar o mapeamento de API `https:// api.example.com/orders/v2/items/categories/5` como uma regra de roteamento e como atualizar os logs de acesso para registrar em log o ID da regra de roteamento que o API Gateway usa para enviar tráfego à sua API.

## Console de gerenciamento da AWS

Como definir o modo de roteamento como `ROUTING_RULE_THEN_API_MAPPING`

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha Custom Domain Names (Nomes de domínios personalizados) no painel de navegação principal.
3. Escolha seu nome de domínio personalizado.
4. Em Detalhes do domínio, escolha Editar.
5. Em Modo de roteamento, escolha `ROUTING_RULE_THEN_API_MAPPING`.
6. Escolha Salvar

Depois de definir o modo de roteamento, crie a regra de roteamento.

Como criar a regra de roteamento

1. Na guia Detalhes do roteamento, escolha Adicionar regra de roteamento.
2. Escolha Adicionar condição e selecione Caminho.
3. Em Caminho, insira **`orders/v2/items/categories/5`**.
4. Em Remover caminho base, escolha Ativo.
5. Em API de destino, escolha sua API de destino.
6. Em Estágio de destino, escolha seu estágio de destino.
7. Escolha Avançar.
8. Em prioridade, insira a prioridade.

Mesmo que você mantiver o mapeamento de API existente, o API Gateway sempre usará a nova regra de roteamento, pois as regras de roteamento sempre têm prioridade sobre os mapeamentos de API.

9. Escolha Salvar alterações.

Depois de criar a regra de roteamento, atualize o formato do log de acesso do estágio ou crie um log para confirmar se o API Gateway usa sua regra de roteamento para enviar tráfego à sua API.

### Como atualizar logs de acesso

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione a API.
3. No painel de navegação principal, selecione Estágios.
4. Em Logs e rastreamento, escolha Editar.

Se você não tiver um grupo de logs, consulte [the section called “Logs do CloudWatch”](#).

5. Adicione **\$context.customDomain.routingRuleIdMatched** ao formato do log.

Esse grupo de logs registra o ID da regra de roteamento que o API Gateway usou para enviar tráfego à sua API. Para obter mais informações, consulte [the section called “Não sei como o API Gateway enviou tráfego às minhas APIs”](#).

6. Escolha Salvar.

Depois de atualizar os logs de acesso, invoque seu nome de domínio personalizado. Veja a seguir um exemplo de comando curl para invocar o nome de domínio personalizado `https://api.example.com` com o caminho base `orders/v2/items/categories/5`.

```
curl "https://api.example.com/orders/v2/items/categories/5"
```

Depois de invocar com sucesso seu nome de domínio personalizado, confirme se o CloudWatch Logs mostra `routingRuleIdMatched`. Para saber como usar o console do CloudWatch Logs para visualizar um grupo de logs, consulte [the section called “Exibir evento de log no console do CloudWatch”](#).

### AWS CLI

1. Use o comando [update-domain-name](#) indicado abaixo para atualizar o nome de domínio `api.example.com` e usar modo de roteamento `ROUTING_RULE_THEN_API_MAPPING`.

```
aws apigatewayv2 update-domain-name \
 --domain-name 'api.example.com' \
 --routing-mode ROUTING_RULE_THEN_API_MAPPING
```

2. Use o comando [create-routing-rule](#) indicado abaixo para criar uma regra de roteamento e recriar o mapeamento de 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. Use o comando [update-stage](#) indicado abaixo para atualizar o formato dos logs de acesso e incluir a variável `$context.customDomain.routingRuleIdMatched`. Essa variável registra o ID da regra de roteamento que o API Gateway usou para enviar tráfego à sua API. Você usa esse log para confirmar se o API Gateway usa sua regra de roteamento para enviar tráfego à sua API. Para obter mais informações, consulte [the section called “Não sei como o API Gateway enviou tráfego às minhas APIs”](#).

```
aws apigateway update-stage \
 --rest-api-id a1bc2c3 \
 --stage-name prod \
 --patch-operations "op=replace,path=/accessLogSettings/format,value='\
$context.path \"$context.customDomain.routingRuleIdMatched \"$context.requestId \
$context.extendedRequestId'"
```

Se você não tiver um grupo de logs, consulte [the section called “Logs do CloudWatch”](#).

4. Use o exemplo de comando curl indicado abaixo para invocar o nome de domínio personalizado com o caminho base `orders/v2/items/categories/5`.

```
curl "https://api.example.com/orders/v2/items/categories/5"
```

5. Use o comando [filter-log-events](#) indicado abaixo para obter os eventos de logs do grupo de logs `access-log-group-orders` que contém o ID da regra de roteamento `abc123`.

```
aws logs filter-log-events --log-group-name access-log-group-orders --filter-pattern abc123
```

Isso confirma que o API Gateway usou a regra de roteamento para enviar tráfego à sua API.

## Solucionar problemas em regras de roteamento

As diretrizes de solução de problemas a seguir podem ajudar a resolver problemas em suas regras de roteamento.

Não sei como o API Gateway enviou tráfego às minhas APIs

Você pode usar logs de acesso para o estágio da API REST a fim de registrar em log e solucionar problemas em suas regras de roteamento. Você pode visualizar o ID da regra de roteamento que o API Gateway usou para enviar tráfego à sua API usando a variável `$context.customDomain.routingRuleIdMatched`. Para visualizar o mapeamento de API que o API Gateway usou para enviar tráfego à sua API, use a variável `$context.customDomain.basePathMatched`.

Para registrar suas regras de roteamento em log, você precisa configurar [um ARN de função do CloudWatch Logs apropriado](#) para sua conta e criar um grupo de logs.

O exemplo de grupo de logs de acesso a seguir pode recuperar as informações relevantes para solucionar problemas em regras de roteamento e mapeamentos de API. O API Gateway preenche somente a variável de contexto do mecanismo de roteamento que ele usa; do contrário, a variável de contexto é -.

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
```

Também recomendamos que você confirme o modo de roteamento do nome de domínio personalizado. Para obter mais informações, consulte [the section called “Definir o modo de roteamento para o nome de domínio personalizado”](#).

Não consigo habilitar as regras de roteamento no meu nome de domínio personalizado

Você pode receber o seguinte erro do 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.
```

Você receberá esse erro se tiver ou tenha tido uma política do IAM que nega acesso ao [BasePathMapping](#) ou ao [ApiMapping](#). Ao habilitar as regras de roteamento para um nome de domínio personalizado, embora sua política continue negando acesso a BasePathMapping ou a ApiMapping, a mesma política pode ser usada para acessar a RoutingRule. Isso pode permitir que um usuário altere o comportamento de roteamento do seu nome de domínio personalizado.

Por exemplo, se você teve uma política como a seguinte:

```
{
 "Sid": "DenyCreatingApiMappings",
 "Effect": "Deny",
 "Action": "apigateway:POST",
 "Resource": [
 "arn:aws:apigateway:us-west-2::/domainnames/example.com/apimappings"
]
}
```

Ao habilitar as regras de roteamento para `example.com`, essa política continuará negando o acesso à criação de um `ApiMapping`, mas não negará o acesso à criação de uma `RoutingRule`.

Recomendamos que você audite as políticas do IAM em sua conta. O seguinte exemplo de política negará acesso à criação de `ApiMapping`, `BasePathMapping` e `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/*"
]
}
```

Depois de confirmar que todas as políticas foram atualizadas, você pode atualizar as configurações da API no nível da conta para habilitar as regras de roteamento para uma região.

Use o comando [update-account](#) indicado abaixo para atualizar as configurações em nível de conta da API para uma região:

```
aws apigateway update-account --patch-operations 'op=remove,path=/
features,value=BlockedForRoutingRules' --region us-west-2
```

Depois de atualizar as configurações em nível de conta da API, você pode alterar o modo de roteamento do nome de domínio personalizado. Você também pode continuar usando as políticas do IAM para negar acesso a `RoutingRules`, `ApiMapping` ou `BasePathMapping`.

Os mapeamentos de API são usados para associar estágios da API a um nome de domínio personalizado para APIs REST.

Você usa mapeamentos de API para conectar estágios de API a um nome de domínio personalizado. Isso encaminha o tráfego às suas APIs por meio do nome de domínio personalizado.

Um mapeamento de API especifica uma API, um estágio e, opcionalmente, um caminho a usar para o mapeamento. Por exemplo, você pode associar `https://api.example.com/orders` ao estágio `production` de uma API.

Você pode mapear os estágios da API HTTP e REST para o mesmo nome de domínio personalizado.

Antes de criar um mapeamento de API, você deve ter uma API, um estágio e um nome de domínio personalizado. Para saber mais sobre como criar um nome de domínio personalizado, consulte [the section called “Configurar um nome de domínio regional personalizado”](#).

Solicitações recebidas em seu nome de domínio personalizado

Quando você associa um nome de domínio personalizado a um estágio da API, o API Gateway remove o caminho base de entrada. Isso remove o caminho base mapeado da invocação para a API. Por exemplo, se o caminho base `https://api.example.com/orders/shop/5` fosse associado ao estágio `test` e você usasse a solicitação `https://api.example.com/orders/shop/5/hats`, o API Gateway invocaria o recurso `/hats` do estágio `test` da API, não o recurso `orders/shop/5/hats`.

Mapear solicitações de API

Veja a seguir como o API Gateway explica os mapeamentos de API.

Você pode criar um mapeamento de API usando mapeamentos de nível único, como um mapeamento de API de `orders` para o estágio `beta` de uma API e um mapeamento de API de `shipping` para o estágio `alpha` de uma API. No caso de nomes de domínio regionais

personalizados com a política de segurança do TLS 1.2, o API Gateway aceita mapeamentos de API de vários níveis. Você pode criar um mapeamento de API de `orders/v1/items` para o estágio alpha de uma API e de `orders/v2/items` para o estágio beta de uma API. Para mapeamentos de API com vários níveis, o API Gateway encaminha as solicitações ao mapeamento de API que tem o prefixo correspondente mais longo.

Você pode criar um mapeamento de API para o caminho vazio (`none`). Se nenhum caminho corresponder à solicitação, o API Gateway enviará a solicitação à API que você associou ao caminho vazio (`none`).

Neste exemplo, o nome de domínio personalizado `https://api.example.com` tem os seguintes mapeamentos de API:

| Mapeamento de API          | API selecionada |
|----------------------------|-----------------|
| (none)                     | API 1           |
| orders                     | API 2           |
| orders/v1/items            | API 3           |
| orders/v2/items            | API 4           |
| orders/v1/items/categories | API 5           |

A tabela a seguir mostra como o API Gateway aplica os mapeamentos de API anteriores a solicitações de exemplo.

| Solicitação                                          | API selecionada | Explicação                                                              |
|------------------------------------------------------|-----------------|-------------------------------------------------------------------------|
| <code>https://api.example.com/orders</code>          | API 2           | A solicitação apresenta correspondência exata a esse mapeamento de API. |
| <code>https://api.example.com/orders/v1/items</code> | API 3           | A solicitação apresenta correspondência exata a esse mapeamento de API. |

| Solicitação                                                       | API selecionada | Explicação                                                                                                                                                                                                                                 |
|-------------------------------------------------------------------|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>https://api.example.com/orders/v2/items</code>              | API 4           | A solicitação apresenta correspondência exata a esse mapeamento de API.                                                                                                                                                                    |
| <code>https://api.example.com/orders/v1/items/123</code>          | API 3           | O API Gateway escolhe o mapeamento com o caminho correspondente mais longo. O 123 no final da solicitação não afeta a seleção. Consulte <a href="#">the section called “Solicitações recebidas em seu nome de domínio personalizado”</a> . |
| <code>https://api.example.com/orders/v2/items/categories/5</code> | API 5           | O API Gateway escolhe o mapeamento com o caminho correspondente mais longo.                                                                                                                                                                |
| <code>https://api.example.com/customers</code>                    | API 1           | O API Gateway usa o mapeamento vazio como um catch-all.                                                                                                                                                                                    |

| Solicitação                                        | API selecionada | Explicação                                                                                                                                                                                                                                                                                                                                                                             |
|----------------------------------------------------|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>https://api.example.com/ordersandmore</code> | API 2           | <p>O API Gateway escolhe o mapeamento com o prefixo correspondente mais longo.</p> <p>Para um nome de domínio personalizado configura do com mapeamentos de nível único, como somente <code>https://api.example.com/orders</code> e <code>https://api.example.com/</code>, o API Gateway escolheria a API 1, pois não há um caminho correspondente com <code>ordersandmore</code>.</p> |

## Restrições

- Em um mapeamento de API, o nome de domínio personalizado e as APIs mapeadas devem estar na mesma conta da AWS.
- Os mapeamentos de API devem conter apenas letras, números e os caracteres a seguir: `$-_.+!*'()/`.
- O comprimento máximo para o caminho em um mapeamento de API é de 300 caracteres.
- É possível ter 200 mapeamentos de API com vários níveis para cada nome de domínio. Esse limite não inclui mapeamento de API com um único nível, como `/prod`.
- Você só pode mapear APIs HTTP para um nome de domínio personalizado regional com a política de segurança TLS 1.2.
- Você não pode mapear APIs WebSocket para o mesmo nome de domínio personalizado que uma API HTTP ou API REST.
- Depois de criar os mapeamentos de API, você deve criar ou atualizar o registro de recursos do provedor de DNS para ser associado ao endpoint da API.

- Se você criar mapeamentos de API com vários níveis, o API Gateway mudará todos os nomes de cabeçalho para minúsculas.

## Crie um mapeamento de API

Para criar um mapeamento de API, você deve primeiro criar um nome de domínio personalizado, uma API e um estágio. O nome de domínio personalizado deve ter um modo de roteamento definido como `ROUTING_RULE_THEN_API_MAPPING` ou `API_MAPPING_ONLY`. Para ter informações sobre como configurar o modo de roteamento, consulte [the section called “Definir o modo de roteamento para o nome de domínio personalizado”](#).

Para obter exemplos de modelos do AWS Serverless Application Model que criam todos os recursos, consulte [Sessões com SAM](#) no GitHub.

## Console de gerenciamento da AWS

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha Custom Domain Names (Nomes de domínios personalizados) no painel de navegação principal.
3. Escolha um nome de domínio personalizado.
4. Na guia Detalhes do roteamento, escolha Configurar mapeamentos de API.
5. Especifique a API, o Estágio e o Caminho do mapeamento.
6. Escolha Salvar.

## AWS CLI

O comando [create-api-mapping](#) a seguir cria um mapeamento de API. Neste exemplo, o API Gateway envia solicitações para `api.example.com/v1/orders` para a API e o estágio especificados.

### Note

Para criar mapeamentos de API com vários níveis, você deve usar `apigatewayv2`.

```
aws apigatewayv2 create-api-mapping \
```

```
--domain-name api.example.com \
--api-mapping-key v1/orders \
--api-id a1b2c3d4 \
--stage test
```

## CloudFormation

O exemplo de CloudFormation a seguir cria um mapeamento de API.

### Note

Para criar mapeamentos de API com vários níveis, você deve usar `AWS::ApiGatewayV2`.

```
MyApiMapping:
 Type: 'AWS::ApiGatewayV2::ApiMapping'
 Properties:
 DomainName: api.example.com
 ApiMappingKey: 'orders/v2/items'
 ApiId: !Ref MyApi
 Stage: !Ref MyStage
```

## Tipos de endereço IP para nomes de domínio personalizados no API Gateway

Ao criar um nome de domínio personalizado, você especifica o tipo de endereços IP que podem invocar seu domínio. É possível escolher IPv4 para resolver endereços IPv4 e invocar seu domínio ou escolher pilha dupla para permitir que endereços IPv4 e IPv6 invoquem seu domínio. Recomendamos que você defina o tipo de endereço IP como pilha dupla para aliviar o esgotamento do espaço IP ou para seu procedimento de segurança. Para ter mais informações sobre os benefícios de um tipo de endereço IP de pilha dupla, consulte [IPv6 na AWS](#).

É possível alterar o tipo de endereço IP atualizando a configuração do endpoint do nome de domínio.

### Considerações sobre tipos de endereço IP

As considerações a seguir podem afetar o uso de tipos de endereço IP.



- O tipo de endereço IP padrão para nomes de domínio personalizados do API Gateway para APIs públicas é IPv4.
- Os nomes de domínio personalizados privados só podem ter um tipo de endereço IP de pilha dupla.
- Seu nome de domínio personalizado não precisa ter o mesmo tipo de endereço IP para todas as APIs associadas a ele. Se você desabilitar seu endpoint de API padrão, isso poderá afetar como os chamadores podem invocar seu domínio.

### Alterar o tipo de endereço IP de um nome de domínio personalizado

É possível alterar o tipo de endereço IP atualizando a configuração do endpoint do nome de domínio. É possível atualizar a configuração do endpoint usando o Console de gerenciamento da AWS, a AWS CLI, o CloudFormation ou um SDK da AWS.

#### Console de gerenciamento da AWS

Como alterar o tipo de endereço IP de um nome de domínio personalizado

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha um nome de domínio personalizado público.
3. Escolha Configurações de endpoint.
4. Em Tipo de endereço IP, escolha IPv4 ou Pilha dupla.
5. Escolha Salvar.

#### AWS CLI

O comando [update-domain-name](#) a seguir atualiza uma API para ter um tipo de endereço IP de pilha dupla:

```
aws apigateway update-domain-name \
 --domain-name dualstack.example.com \
 --patch-operations "op='replace',path='/endpointConfiguration/
ipAddressType',value='dualstack'"
```

A saída será exibida da seguinte forma:

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

## Escolher uma política de segurança para o domínio personalizado no API Gateway

Uma política de segurança é uma combinação predefinida da versão mínima do TLS e dos pacotes de criptografia oferecida pelo Amazon API Gateway. Quando seus clientes estabelecem um handshake do TLS para a API ou o nome do domínio personalizado, a política de segurança aplica a versão do TLS e o pacote de criptografia aceitos pelo API Gateway. As políticas de segurança protegem suas APIs e nomes de domínio personalizados contra problemas de segurança de rede, como violação e espionagem entre um cliente e o servidor.

O API Gateway aceita políticas de segurança legadas e políticas de segurança aprimoradas. TLS\_1\_0 e TLS\_1\_2 são políticas de segurança legadas. Use essas políticas de segurança para workloads generalizadas ou para começar a criar uma API. Qualquer política que comece com SecurityPolicy\_ é uma política de segurança aprimorada. Use essas políticas para workloads reguladas, governança avançada ou para usar criptografia pós-quântica. Ao usar uma política de segurança aprimorada, você também deve definir o modo de acesso ao endpoint para governança adicional. Para obter mais informações, consulte [the section called “Modo de acesso ao endpoint”](#).

### Considerações

Veja a seguir algumas considerações sobre políticas de segurança para nomes de domínio personalizados para APIs REST no API Gateway:

- Você não pode habilitar o TLS mútuo em um nome de domínio que utilize uma política de segurança aprimorada.

- Não é possível associar uma API HTTP a um nome de domínio que use uma política de segurança aprimorada.
- Se você habilitar o mapeamento de caminho de base de vários níveis a uma API REST que utilize uma política de segurança aprimorada, não poderá criar um mapeamento de caminho de base para uma API HTTP com o mesmo nome de domínio.
- Sua API pode ser associada a um nome de domínio personalizado com uma política de segurança diferente da política da sua API. Quando você invoca esse nome de domínio personalizado, o API Gateway utiliza a política de segurança da API para negociar o handshake TLS. Se você desabilitar seu endpoint de API padrão, isso poderá afetar como os chamadores podem invocar sua API.
- O API Gateway aceita políticas de segurança em todas as APIs. No entanto, você só pode escolher uma política de segurança para as APIs REST. O API Gateway comporta apenas a política de segurança TLS\_1\_2 para APIs HTTP ou de WebSocket.
- O API Gateway não comporta a atualização de uma política de segurança para um nome de domínio com vários tipos de endpoint. Caso você tenha vários tipos de endpoint para um nome de domínio, exclua um deles para atualizar a política de segurança.

### Como o API Gateway aplica políticas de segurança

O exemplo a seguir mostra como o API Gateway aplica políticas de segurança utilizando a política `SecurityPolicy_TLS13_1_3_2025_09` como exemplo.

A política de segurança `SecurityPolicy_TLS13_1_3_2025_09` aceita tráfego TLS 1.3 e rejeita tráfego TLS 1.2 e TLS 1.0. Para o tráfego TLS 1.3, a política de segurança aceita os seguintes pacotes de criptografia:

- `TLS_AES_128_GCM_SHA256`
- `TLS_AES_256_GCM_SHA384`
- `TLS_CHACHA20_POLY1305_SHA256`

O API Gateway não aceita nenhum outro pacote de criptografia. Por exemplo, a política de segurança rejeitaria qualquer tráfego TLS 1.3 que utilize o pacote de criptografia AES128-SHA.

Para monitorar quais protocolos TLS e criptografias os clientes usaram para acessar seu API Gateway, você pode usar as variáveis de contexto `$context.tlsVersion` e

`$context.cipherSuite` em seus logs de acesso. Para obter mais informações, consulte [the section called “Monitorar”](#).

Para ver as políticas de segurança padrão para todas as APIs REST e nomes de domínio personalizados, consulte [the section called “Políticas de segurança padrão”](#). Para ver as políticas de segurança aceitas para todas as APIs REST e nomes de domínio personalizados, consulte [the section called “Políticas de segurança aceitas”](#).

Alterar a política de segurança do seu nome de domínio personalizado

Se você alterar sua política de segurança, a atualização levará cerca de 15 minutos para ser concluída. É possível monitorar o `lastUpdateStatus` de seu nome de domínio personalizado. No decorrer da atualização do seu nome de domínio personalizado, o `lastUpdateStatus` será `PENDING` e, quando for concluído, será `AVAILABLE`.

Quando você usa uma política de segurança que comece com `SecurityPolicy_`, também deve ativar o modo de acesso ao endpoint. Para obter mais informações, consulte [the section called “Modo de acesso ao endpoint”](#).

Console de gerenciamento da AWS

Como alterar a política de segurança de um nome de domínio personalizado

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha um nome de domínio personalizado que envie tráfego para as APIs REST.

Verifique se há apenas um tipo de endpoint associado ao seu nome de domínio personalizado.

3. Escolha Configurações de nome de domínio personalizado e, depois, selecione Editar.
4. Em Política de segurança, selecione uma nova política.
5. Em Modo de acesso ao endpoint, escolha Restrito.
6. Escolha Salvar alterações.

AWS CLI

O seguinte comando [update-domain-name](#) atualiza um nome de domínio para utilizar a política de segurança `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"
 }
]'
```

A saída será exibida da seguinte forma:

```
{
 "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"
}
```

## Informações sobre APIs HTTP e APIs de WebSocket

Para saber mais sobre APIs HTTP e APIs de WebSocket, consulte [the section called “Política de segurança para APIs HTTP no API Gateway”](#) e [the section called “Política de segurança para APIs de WebSocket no API Gateway”](#).

## Desabilitar o endpoint padrão para APIs REST

Por padrão, os clientes podem invocar sua API usando o endpoint `execute-api` gerado pelo API Gateway para sua API. Para garantir que os clientes possam acessar sua API somente usando um nome de domínio personalizado, desabilite o endpoint `execute-api` padrão. Os clientes ainda podem se conectar ao endpoint padrão, mas receberão um código de status 403 `Forbidden`. A desabilitação do endpoint padrão afeta todos os estágios da API. Essa configuração entra em vigor

quando você atualiza qualquer configuração em qualquer estágio, como atualizar a implantação no estágio.

O procedimento a seguir mostra como desabilitar o endpoint padrão de uma API REST.

### Console de gerenciamento da AWS

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST.
3. No painel de navegação principal, selecione Configurações da API.
4. Escolha uma API.
5. Em Detalhes da API, escolha Editar.
6. Em Endpoint padrão, selecione Inativo.
7. Escolha Salvar alterações.
8. No painel de navegação principal, selecione Recursos.
9. Escolha Implantar API.
10. Reimplante sua API em um estágio ou atualize qualquer configuração em um estágio para que a atualização entre em vigor.

### AWS CLI

O comando [update-rest-api](#) indicado abaixo desabilita o endpoint padrão:

```
aws apigateway update-rest-api \
 --rest-api-id abcdef123 \
 --patch-operations op=replace,path=/disableExecuteApiEndpoint,value='True'
```

Depois de desabilitar o endpoint padrão, é necessário implantar sua API para que a alteração entre em vigor.

O comando [create-deployment](#) indicado abaixo cria uma implantação e a associa a um estágio:

```
aws apigateway create-deployment \
 --rest-api-id abcdef123 \
 --stage-name dev
```

## Configurar verificações de integridade personalizadas para failover de DNS para um API Gateway

É possível usar as verificações de integridade do Amazon Route 53 para controlar o failover de DNS de uma API do API Gateway em uma Região da AWS primária para outra em uma região secundária. Isso pode ajudar a mitigar os impactos no caso de um problema regional. Se você usa um domínio personalizado, pode realizar o failover sem exigir que os clientes alterem os endpoints da API.

Quando você escolhe [Evaluate Target Health](#) (Avaliar integridade do destino) para um registro de alias, esses registros falham somente quando o serviço API Gateway não está disponível na região. Em alguns casos, suas próprias APIs do API Gateway podem sofrer interrupções antes desse período. Para controlar o failover de DNS diretamente, configure verificações de integridade personalizadas do Route 53 para as APIs do API Gateway. Neste exemplo, você usa um alarme do CloudWatch que ajuda os operadores a controlar o failover de DNS. Para ver mais exemplos e outras considerações ao configurar o failover, consulte [Criar mecanismos de recuperação de desastres usando o Route 53](#) e [Realizar verificações de integridade do Route 53 em recursos privados em uma VPC com o AWS Lambda e o CloudWatch](#).

### Tópicos

- [Pré-requisitos](#)
- [Etapa 1: configurar recursos](#)
- [Etapa 2: iniciar o failover para a região secundária](#)
- [Etapa 3: testar o failover](#)
- [Etapa 4: retornar à região primária](#)
- [Próximas etapas: personalizar e testar regularmente](#)

### Pré-requisitos

Para concluir esse procedimento, você deve criar e configurar os seguintes recursos:

- Um nome de domínio de sua propriedade.
- Um certificado do ACM para esse nome de domínio em duas Regiões da AWS. Para obter mais informações, consulte [the section called “Pré-requisitos referentes a nomes de domínio personalizados”](#).

- Uma zona hospedada no Route 53 para seu nome de domínio. Para obter mais informações, consulte [Trabalhar com zonas hospedadas](#) no Guia do desenvolvedor do Amazon Route 53.

Para receber mais informações sobre como criar registros DNS de failover do Route 53 para os nomes de domínio, consulte [Escolher uma política de roteamento](#) no Guia do desenvolvedor do Amazon Route 53. Para receber mais informações sobre como monitorar um alarme do CloudWatch, consulte [Monitorar um alarme do CloudWatch](#) no Guia do desenvolvedor do Amazon Route 53.

## Etapa 1: configurar recursos

Neste exemplo, você cria os seguintes recursos para configurar o failover de DNS para seu nome de domínio:

- APIs do API Gateway em duas Regiões da AWS
- Nomes de domínio personalizados do API Gateway com o mesmo nome em duas Regiões da AWS
- Mapeamentos de API do API Gateway que conectam as APIs do API Gateway aos nomes de domínio personalizados
- Registros de failover de DNS do Route 53 para os nomes de domínio
- Um alarme do CloudWatch na região secundária
- Uma verificação de integridade do Route 53 com base no alarme do CloudWatch na região secundária

Primeiro, certifique-se de que você tenha todos os recursos necessários nas regiões primária e secundária. A região secundária deve conter o alarme e a verificação de integridade. Dessa maneira, você não depende da região primária para realizar o failover. Por exemplo, modelos do CloudFormation que criam esses recursos, consulte [primary.yaml](#) e [secondary.yaml](#).

### Important

Antes do failover para a região secundária, certifique-se de que todos os recursos necessários estejam disponíveis. Caso contrário, a API não estará pronta para o tráfego na região secundária.



## Etapa 2: iniciar o failover para a região secundária

No exemplo a seguir, a região em espera recebe uma métrica do CloudWatch e inicia o failover. Usamos uma métrica personalizada que exige a intervenção do operador para iniciar o failover.

```
aws cloudwatch put-metric-data \
 --metric-name Failover \
 --namespace HealthCheck \
 --unit Count \
 --value 1 \
 --region us-west-1
```

Substitua os dados de métrica pelos dados correspondentes para o alarme do CloudWatch que você configurou.

## Etapa 3: testar o failover

Invoque a API e verifique se você recebeu uma resposta da região secundária. Se você usou os modelos de exemplo na etapa 1, a resposta muda de {"message": "Hello from the primary Region!"} para {"message": "Hello from the secondary Region!"} após o failover.

```
curl https://my-api.example.com

{"message": "Hello from the secondary Region!"}
```

## Etapa 4: retornar à região primária

Para retornar à região primária, envie uma métrica do CloudWatch que resulte em aprovação na verificação de integridade.

```
aws cloudwatch put-metric-data \
 --metric-name Failover \
 --namespace HealthCheck \
 --unit Count \
 --value 0 \
 --region us-west-1
```

Substitua os dados de métrica pelos dados correspondentes para o alarme do CloudWatch que você configurou.

Invoque a API e verifique se você recebeu uma resposta da região primária. Se você usou os modelos de exemplo na etapa 1, a resposta muda de {"message": "Hello from the secondary Region!"} para {"message": "Hello from the primary Region!"}.

```
curl https://my-api.example.com
```

```
{"message": "Hello from the primary Region!"}
```

Próximas etapas: personalizar e testar regularmente

Este exemplo demonstra uma maneira de configurar o failover de DNS. É possível usar uma variedade de métricas do CloudWatch ou endpoints HTTP para as verificações de integridade que gerenciam o failover. Teste regularmente os mecanismos de failover para garantir que eles funcionem conforme o esperado e que os operadores estejam familiarizados com seus procedimentos de failover.

## Otimizar o desempenho das APIs REST

Depois de disponibilizar sua API para ser chamada, você pode perceber que ela precisa ser otimizada para melhorar a capacidade de resposta. O API Gateway fornece algumas estratégias para otimizar sua API, como cache de resposta e compactação de carga. Nesta seção, você pode aprender como habilitar esses recursos.

### Tópicos

- [Configurações de cache para APIs REST no API Gateway](#)
- [Compressão de carga útil para APIs REST no API Gateway](#)

## Configurações de cache para APIs REST no API Gateway

É possível habilitar o armazenamento em cache da API no API Gateway para armazenar em cache as respostas do endpoint. Com o armazenamento em cache, você pode reduzir o número de chamadas feitas para o endpoint e também melhorar a latência de solicitações para a sua API.

Quando o armazenamento em cache é habilitado para um estágio, o API Gateway armazena em cache as respostas do seu endpoint por um período Time-to-live (TTL – vida útil) especificado, em segundos. Depois, o API Gateway responde à solicitação examinando a resposta do endpoint no cache em vez de fazer uma solicitação ao seu endpoint. O valor de TTL padrão para o

armazenamento em cache de APIs é de 300 segundos. O valor de TTL máximo é de 3600 segundos. TTL=0 significa que o armazenamento em cache está desabilitado.

#### Note

O armazenamento em cache é o melhor esforço. Você pode usar as métricas CacheHitCount e CacheMissCount no Amazon CloudWatch para monitorar solicitações que o API Gateway atende pelo cache da API.

O tamanho máximo de uma resposta que pode ser armazenada em cache é 1.048.576 bytes. A criptografia de dados de cache pode aumentar o tamanho da resposta quando está sendo armazenada em cache.

Este é um serviço qualificado da HIPAA. Para obter mais informações sobre a AWS, a Lei de Portabilidade e Responsabilidade de Seguro de Saúde de 1996 dos EUA (HIPAA) e o uso dos serviços da AWS para processar, armazenar e transmitir informações de saúde protegidas (PHI), consulte [Visão geral da HIPAA](#).

#### Important

Ao habilitar o armazenamento em cache para um estágio, somente métodos GET têm o armazenamento em cache habilitado por padrão. Isso ajuda a garantir a segurança e a disponibilidade da sua API. Você pode habilitar o armazenamento em cache para outros métodos, [substituindo as configurações do método](#).

#### Important

O armazenamento em cache é cobrado por hora com base no tamanho do cache escolhido. O armazenamento em cache não é elegível para o nível gratuito da AWS. Para obter mais informações, consulte [Preços do API Gateway](#).

## Habilitar o armazenamento em cache do Amazon API Gateway

No API Gateway, é possível habilitar o armazenamento em cache de um estágio específico.

Ao habilitar o armazenamento em cache, você deve escolher uma capacidade de cache. Em geral, uma capacidade maior proporciona melhor desempenho, mas também custa mais. Para ver os tamanhos de cache compatíveis, consulte [cacheClusterSize](#) na Referência de API do API Gateway.

O API Gateway habilita o armazenamento em cache por meio da criação de uma instância de cache dedicada. Esse processo pode demorar até 4 minutos.

O API Gateway altera a capacidade de armazenamento em cache removendo a instância de cache existente e criando uma nova com capacidade modificada. Todos os dados armazenados em cache existentes são excluídos.

 Note

A capacidade do cache afeta a CPU, a memória e a largura de banda da rede da instância de cache. Como resultado, a capacidade do cache pode afetar a performance do cache. O API Gateway recomenda que você execute um teste de carga de 10 minutos para verificar se a capacidade do cache é apropriada para sua carga de trabalho. Certifique-se de que o tráfego durante o teste de carga corresponde ao tráfego da produção. Por exemplo, inclua padrões de tráfego crescente, constante e em picos. O teste de carga deve incluir respostas que podem ser servidas a partir do cache, bem como respostas exclusivas que adicionam itens ao cache. Monitore as métricas de latência, 4xx, 5xx, acertos de cache e erros de cache durante o teste de carga. Ajuste a capacidade do cache conforme necessário com base nessas métricas. Para obter mais informações sobre o teste de carga, consulte [Como seleciono a melhor capacidade de cache do Amazon API Gateway para não atingir um limite de taxa?](#).

## Console de gerenciamento da AWS

No console do API Gateway, configure o armazenamento em cache na página Estágios. Provisione o cache do estágio e especifique uma configuração padrão de cache no nível de método. Se você ativar o cache padrão no nível de método, o armazenamento em cache no nível de método será ativado para todos os métodos GET no estágio, a menos que esse método tenha uma substituição de método. Todos os métodos GET adicionais que você implantar em seu estágio terão cache no nível de método. Para definir a configuração do armazenamento em cache no nível de método para métodos específicos do seu estágio, você pode usar substituições de método. Para saber mais sobre substituições de método, consulte [the section called “Substituir o armazenamento em cache de estágios para o armazenamento em cache de métodos”](#).

Para configurar o armazenamento em cache de API para um determinado estágio:

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha Stages (Estágios).
3. Na lista Stages (Estágios) da API, escolha o estágio.
4. Na seção Detalhes do estágio, selecione Editar.
5. Em Configurações adicionais, para Configurações de cache, ative Provisionar cache de APIs.

Isso provisiona um cluster de cache para seu estágio.

6. Para ativar o armazenamento em cache para seu estágio, ative Armazenamento em cache padrão no nível de método.

Isso ativa o armazenamento em cache no nível de método para todos os métodos GET em seu estágio. Todos os métodos GET adicionais que você implantar nesse estágio terão um cache no nível de método.

 Note

Se você já tiver uma configuração para um cache no nível de método, alterar a configuração de armazenamento em cache padrão no nível de método não afetará essa configuração existente.

### Additional settings

#### Cache settings [Info](#)

You can enable API caching to cache your endpoint's responses. With caching, you can reduce the number of calls made to your endpoint and also improve the latency of requests to your API. Caching is charged by the hour based on cache size, see API Gateway pricing for details.

- ☒ **Provision API cache**  
Provision API caching capabilities for your stage. Caching is not active until you enable the method-level cache.
- ☒ **Default method-level caching**  
Activate method-level caching for all GET methods in this stage.

7. Escolha Salvar alterações.

## AWS CLI

O comando [update-stage](#) indicado abaixo atualiza um estágio para provisionar um cache e ativa o armazenamento em cache em nível de método para todos os métodos GET em seu estágio:

```
aws apigateway update-stage \
 --rest-api-id a1b2c3 \
 --stage-name 'prod' \
 --patch-operations file://patch.json
```

O conteúdo de `patch.json` é o seguinte:

```
[
 {
 "op": "replace",
 "path": "/cacheClusterEnabled",
 "value": "true"
 },
 {
 "op": "replace",
 "path": "/cacheClusterSize",
 "value": "0.5"
 },
 {
 "op": "replace",
 "path": "/*/*/caching/enabled",
 "value": "true"
 }
]
```

### Note

Se você já tiver uma configuração para um cache no nível de método, alterar a configuração de armazenamento em cache padrão no nível de método não afetará essa configuração existente.

### Note

A criação ou exclusão de um cache leva cerca de 4 minutos para ser concluída pelo API Gateway.

Quando um cache é criado, o valor de Cluster de cache é alterado de `Create in progress` para `Active`. Quando a exclusão do cache é concluída, o valor de Cluster de cache é alterado de `Delete in progress` para `Inactive`.

Quando você ativa o cache no nível de método para todos os métodos em seu estágio, o valor de Armazenamento em cache padrão no nível de método é alterado para `Active`. Se você desativar o cache no nível de método para todos os métodos em seu estágio, o valor de Armazenamento em cache padrão no nível de método será alterado para `Inactive`. Se você tiver uma configuração para um cache no nível de método, alterar o status do cache não afetará essa configuração.

Ao habilitar o armazenamento em cache dentro das Configurações de cache de um estágio, somente métodos GET são armazenados em cache. Para garantir a segurança e a disponibilidade da sua API, recomendamos não alterar essa configuração. No entanto, você pode habilitar o armazenamento em cache para outros métodos, [substituindo as configurações do método](#).

Se quiser verificar se o armazenamento em cache está funcionando como esperado, você tem duas opções gerais:

- Inspecionar as métricas do CloudWatch de `CacheHitCount` e `CacheMissCount` para a sua API e estágio.
- Colocar um carimbo de data/hora na resposta.

 **Note**

Não use o cabeçalho `X-Cache` da resposta do CloudFront para determinar se a sua API está sendo atendida pela instância de cache do API Gateway.

## Substituir o armazenamento em cache no nível de estágio do API Gateway para armazenamento em cache no nível de método

Você pode substituir as configurações de cache no nível de estágio ativando ou desativando o armazenamento em cache para um método específico. Também é possível modificar o período TTL ou ativar e desativar a criptografia para respostas armazenadas em cache.

Se houver previsão de que determinado método armazenado em cache receberá dados confidenciais nas respectivas respostas, criptografe os dados de cache. Você pode precisar fazer isso para cumprir vários requisitos de conformidade. Para ter mais informações, consulte [Amazon API Gateway controls](#) no Guia do usuário do AWS Security Hub.

## Console de gerenciamento da AWS

Se você alterar a configuração do armazenamento em cache padrão no nível de método nos Detalhes do estágio, isso não afetará as configurações de cache no nível de método que têm substituições.

Se você antecipar que um determinado método armazenado em cache receberá dados confidenciais em suas respostas, em Cache Settings (Configurações de cache), escolha Encrypt cache data (Criptografar dados de cache).

Para configurar o armazenamento em cache de API para métodos individuais usando o console:

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione a API.
3. Escolha Stages (Estágios).
4. Na lista Stages (Estágios) da API, expanda o estágio e escolha um método na API.
5. Na seção Substituições de método, escolha Editar.
6. Na seção Configurações do método, ative ou desative a opção Ativar cache de método ou personalize qualquer outra opção desejada.

### Note

O armazenamento em cache só estará ativo quando você provisionar um cluster de cache para seu estágio.

7. Escolha Salvar.

## AWS CLI

O comando [update-stage](#) indicado abaixo desativa o cache somente para o método GET /pets:

```
aws apigateway update-stage /
 --rest-api-id a1b2c3 /
```



```
--stage-name 'prod' /
--patch-operations file://patch.json
```

O conteúdo de `patch.json` é o seguinte:

```
[{
 "op": "replace",
 "path": "/~1pets/GET/caching/enabled",
 "value": "false"
}]
```

## Usar parâmetros de método ou integração como chaves de cache para indexar respostas em cache

É possível usar um método ou parâmetro de integração como chaves de cache para indexar respostas armazenadas em cache. Isso inclui cabeçalhos personalizados, caminhos de URL ou strings de consulta. É possível especificar alguns ou todos esses parâmetros como chave de cache, mas você deve especificar pelo menos um valor. Quando você tem uma chave de cache, o API Gateway armazena em cache as respostas de cada valor de chave separadamente, inclusive quando a chave de cache não está presente.

### Note

As chaves de cache são necessárias ao configurar o armazenamento em cache em um recurso.

Por exemplo, suponha que você tenha uma solicitação no seguinte formato:

```
GET /users?type=... HTTP/1.1
host: example.com
...
```

Nessa solicitação, `type` pode ter um valor de `admin` ou `regular`. Se você incluir o parâmetro `type` como parte da chave do cache, as respostas de `GET /users?type=admin` serão armazenadas em cache separadamente daquelas de `GET /users?type=regular`.

Quando uma solicitação de método ou integração usa mais de um parâmetro, você pode optar por incluir alguns ou todos os parâmetros para criar a chave de cache. Por exemplo, você pode incluir apenas o parâmetro `type` na chave de cache para a seguinte solicitação, feita na ordem listada dentro de um período de TTL:

```
GET /users?type=admin&department=A HTTP/1.1
host: example.com
...
```

A resposta dessa solicitação é armazenada em cache e usada para atender à seguinte solicitação:

```
GET /users?type=admin&department=B HTTP/1.1
host: example.com
...
```

## Console de gerenciamento da AWS

Para incluir um parâmetro de solicitação de método ou integração como parte de uma chave de cache no console do API Gateway, selecione **Caching** (Armazenamento em cache) depois de adicionar o parâmetro.

## Edit method request

### Method request settings

#### Authorization

None

#### Request validator

None

☐ API key required

#### Operation name - optional

GetPets

### ▼ URL query string parameters

#### Name

page

#### Required

☐

#### Caching

☒

Remove

type

☐☐

Remove

Add query string

## AWS CLI

O comando [put-method](#) indicado abaixo cria um método GET e exige o parâmetro de string 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"
```

O comando [put-integration](#) indicado abaixo cria uma integração para o método GET com um endpoint HTTP e especifica que o API Gateway armazene em cache o parâmetro de solicitação do 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 o API Gateway armazene em cache um parâmetro de solicitação de integração, use `integration.request.location.name` como o parâmetro da chave de cache.

## Liberar o cache de estágio de APIs no API Gateway

Quando o armazenamento em cache de APIs está habilitado, você pode liberar o cache do estágio de API para garantir que os clientes da API obtenham as respostas mais recentes dos endpoints de integração.

### Console de gerenciamento da AWS

#### Liberar o cache de estágios de API

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API que tenha um estágio com cache.
3. No painel de navegação principal, selecione Estágios e escolha seu estágio com cache.
4. Escolha o menu Ações de estágio e selecione Liberar cache do estágio.

### AWS CLI

O comando [flush-stage-cache](#) indicado abaixo libera o cache do estágio:

```
aws apigateway flush-stage-cache \
 --rest-api-id a1b2c3 \
 --stage-name prod
```

#### Note

Após o cache ser enviado, as respostas serão atendidas no endpoint de integração até que o cache seja criado novamente. Durante esse período, o número de solicitações enviadas

ao endpoint de integração poderá aumentar. Isso pode aumentar temporariamente a latência geral da sua API.

## Invaldar uma entrada de cache do API Gateway

Um cliente da sua API pode invalidar uma entrada de cache existente e recarregá-la no endpoint de integração para solicitações individuais. O cliente deve enviar uma solicitação que contenha o cabeçalho `Cache-Control: max-age=0`. O cliente recebe a resposta diretamente do endpoint de integração em vez do cache, desde que o cliente esteja autorizado a fazer isso. Isso substitui a entrada de cache existente pela nova resposta, que é obtida do endpoint de integração.

Para conceder permissão a um cliente, anexe uma política com o formato a seguir a uma função de execução do IAM para o usuário.

### Note

Não há suporte à invalidação de cache entre contas.

## 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"
]
 }
]
}
```

Essa política permite que o serviço de execução do API Gateway invalide o cache para solicitações referentes aos recursos especificados. Para especificar um grupo de recursos direcionados, use um caractere curinga (\*) para `account-id`, `api-id` e outras entradas no valor de ARN de Resource. Para obter mais informações sobre como definir permissões para o serviço de execução do API Gateway, consulte [Controlar o acesso a uma API REST com permissões do IAM](#).

Se você não impuser uma política `InvalidateCache` (ou marcar a caixa de seleção `Require authorization` (Exigir autorização) no console), qualquer cliente poderá invalidar o cache da API. Se a maioria ou todos os clientes invalidarem o cache de API, isso poderá aumentar significativamente a latência da sua API.

Quando a política está em vigor, o armazenamento em cache está habilitado e a autorização é necessária.

Você pode especificar como o API Gateway lida com solicitações não autorizadas escolhendo entre as seguintes opções:

Recusar a solicitação com o código de status 403

O API Gateway retorna uma resposta 403 `Unauthorized`.

Para definir essa opção usando a API, use `FAIL_WITH_403`.

Ignorar cabeçalho de controle de cache; adicionar um aviso no cabeçalho de resposta

O API Gateway processa a solicitação e adiciona um cabeçalho de aviso na resposta.

Para definir essa opção usando a API, use `SUCCEED_WITH_RESPONSE_HEADER`.

Ignorar cabeçalho de controle de cache

O API Gateway processa a solicitação e não adiciona um cabeçalho de aviso na resposta.

Para definir essa opção usando a API, use `SUCCEED_WITHOUT_RESPONSE_HEADER`.

Você pode definir o comportamento de tratamento de solicitações não autorizadas usando o console do API Gateway ou a AWS CLI.

Console de gerenciamento da AWS

Como especificar de que forma as solicitações não autorizadas são tratadas

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.

2. Escolha uma API que tenha um estágio com cache.
3. No painel de navegação principal, selecione Estágios e escolha seu estágio com cache.
4. Em Detalhes do estágio, escolha Editar.
5. Em Tratamento de solicitações não autorizadas, selecione uma opção.
6. Escolha Continuar.
7. Revise as alterações e selecione Salvar alterações.

## AWS CLI

O comando [update-stage](#) indicado abaixo atualiza um estágio para lidar com solicitações não autorizadas recusando a solicitação com o código de status 403:

```
aws apigateway update-stage /
 --rest-api-id a1b2c3 /
 --stage-name 'prod' /
 --patch-operations 'op=replace,path=/**/caching/
 unauthorizedCacheControlHeaderStrategy,value="FAIL_WITH_403"'
```

## Exemplo do CloudFormation de um estágio com cache

O modelo CloudFormation a seguir cria um exemplo de API, provisiona um cache de 0.5 GB para o estágio Prod e ativa o cache em nível de método para todos os métodos GET.

### Important

O armazenamento em cache é cobrado por hora com base no tamanho do cache escolhido. O armazenamento em cache não é elegível para o nível gratuito da AWS. Para obter mais informações, consulte [Preços do API Gateway](#).

## Exemplo de modelo CloudFormation

```
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
```

## Compressão de carga útil para APIs REST no API Gateway

O API Gateway permite que o cliente chame uma API com cargas compactadas usando uma das [codificações de conteúdo compatíveis](#). Por padrão, o API Gateway oferece suporte à



descompactação da carga de solicitação do método. No entanto, você deve configurar sua API para habilitar a compactação da carga de resposta do método.

Para habilitar a compactação em uma [API](#), defina a propriedade `minimumCompressionsSize` como um inteiro não negativo entre 0 e 10485760 (10 milhões de bytes) ao criar a API ou depois de criá-la. Para desabilitar a compactação na API, defina `minimumCompressionSize` como nulo ou remova-o por completo. É possível habilitar ou desabilitar a compactação de uma API usando o console do API Gateway, a AWS CLI ou a API REST do API Gateway.

Se você deseja que a compactação seja aplicada em cargas de qualquer tamanho, defina o valor de `minimumCompressionSize` como zero. No entanto, a compactação de dados de um volume pequeno pode, na verdade, aumentar o volume final dos dados. Além disso, a compactação no API Gateway e a descompactação no cliente podem aumentar a latência geral e exigir mais tempo de computação. Você deve executar casos de teste com a sua API para determinar um valor ideal.

O cliente pode enviar uma solicitação de API com uma carga compactada e um cabeçalho `Content-Encoding` apropriado para que o API Gateway descompacte e aplique os modelos de mapeamento adequados, antes de passar a solicitação para o endpoint de integração. Depois que a compactação for habilitada e a API for implantada, o cliente poderá receber uma resposta da API com uma carga compactada se ela especificar um cabeçalho `Accept-Encoding` apropriado na solicitação do método.

Quando o endpoint de integração espera e retorna cargas JSON não compactadas, qualquer modelo de mapeamento que esteja configurado para uma carga JSON não compactada será aplicável para a carga compactada. Para uma carga de solicitação de método compactada, o API Gateway descompacta a carga, aplica o modelo de mapeamento e passa a solicitação mapeada para o endpoint de integração. Para uma carga de resposta de integração não compactada, o API Gateway aplica o modelo de mapeamento, compacta a carga mapeada e retorna a carga compactada para o cliente.

## Tópicos

- [Habilitar a compactação de payload para uma API no API Gateway](#)
- [Chamar um método de API com uma payload compactada no API Gateway](#)
- [Receber uma resposta da API com uma payload compactada no API Gateway](#)

## Habilitar a compactação de payload para uma API no API Gateway

É possível habilitar a compactação para uma API usando o console do API Gateway, a AWS CLI ou um SDK da AWS.

Para uma API existente, implante a API depois de habilitar a compactação, para que a alteração entre em vigor. Para uma nova API, você pode implantar a API depois que a configuração da API for concluída.

### Note

A codificação de conteúdo com a prioridade mais alta deve ser aquela compatível com o API Gateway. Se não for, a compactação não será aplicada à carga da resposta.

### Tópicos

- [Habilitar a compactação de carga para uma API usando o console do API Gateway](#)
- [Habilitar a compactação de carga útil para uma API usando a AWS CLI](#)
- [Codificação de conteúdo compatível com o API Gateway](#)

### Habilitar a compactação de carga para uma API usando o console do API Gateway

O procedimento a seguir descreve como habilitar a compactação de carga para uma API.

#### Como habilitar a compactação de carga usando o console do API Gateway

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API existente ou crie uma nova.
3. No painel de navegação principal, selecione Configurações da API.
4. Na seção Detalhes da API, escolha Editar.
5. Ative a Codificação de conteúdo para habilitar a compactação da carga útil. Em Tamanho mínimo do corpo, insira um número para o tamanho da compactação (em bytes). Para desativar a compactação, desative a opção Codificação de conteúdo.
6. Escolha Salvar alterações.

### Habilitar a compactação de carga útil para uma API usando a AWS CLI

O comando [create-rest-api](#) indicado abaixo cria uma API com compactação de carga útil:

```
aws apigateway create-rest-api \
 --name "My test API" \
 --minimum-compression-size 0
```

O comando [update-rest-api](#) indicado abaixo habilita a compactação de carga útil para uma API existente:

```
aws apigateway update-rest-api \
 --rest-api-id 1234567890 \
 --patch-operations op=replace,path=/minimumCompressionSize,value=0
```

A propriedade `minimumCompressionSize` tem um valor inteiro não negativo entre 0 e 10485760 (10 milhões de bytes). Ela mede o limite de compactação. Se o tamanho da carga útil é menor do que esse valor, a compactação ou descompactação não são aplicadas na carga. Ao configurar para zero, a compactação pode ser definida para qualquer tamanho da carga útil.

O comando [update-rest-api](#) indicado abaixo desativa a compactação de carga útil:

```
aws apigateway update-rest-api \
 --rest-api-id 1234567890 \
 --patch-operations op=replace,path=/minimumCompressionSize,value=
```

Você também pode definir `value` como uma string vazia `""` ou omitir a propriedade `value` completamente na chamada anterior.

## Codificação de conteúdo compatível com o API Gateway

O API Gateway é compatível com as seguintes codificações de conteúdo:

- `deflate`
- `gzip`
- `identity`

O API Gateway também é compatível com o formato de cabeçalho `Accept-Encoding` a seguir, de acordo com a especificação [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

## Chamar um método de API com uma payload compactada no API Gateway

Para fazer uma solicitação da API com uma carga compactada, o cliente deve definir o cabeçalho Content-Encoding com uma das [codificações de conteúdo compatíveis](#).

Suponha que você seja um cliente de API e queira chamar o método de API PetStore (POST /pets). Não chame o método usando esta saída JSON:

```
POST /pets
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
Content-Length: ...

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

Em vez disso, você pode chamar o método com a mesma carga compactada usando a codificação GZIP:

```
POST /pets
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
Content-Encoding:gzip
Content-Length: ...

    RPP*  ,HU  RPJ  0W    e&    L,  ,-y  j
```

Quando o API Gateway recebe a solicitação, ele verifica se a codificação de conteúdo especificada é compatível. Em seguida, ele tenta descompactar a carga com a codificação de conteúdo especificada. Se a descompactação for bem-sucedida, ele enviará a solicitação para o endpoint de integração. Se a codificação especificada não for compatível ou a carga fornecida não estiver compactada, o API Gateway retornará uma resposta com o erro 415 `Unsupported Media Type`. O erro não será registrado em log no CloudWatch Logs se ele ocorrer na fase inicial de descompactação antes de sua API e etapa serem identificadas.

## Receber uma resposta da API com uma payload compactada no API Gateway

Ao fazer uma solicitação em uma API habilitada para compactação, o cliente pode optar por receber uma carga de resposta compactada de um determinado formato especificando um cabeçalho `Accept-Encoding` com uma [codificação de conteúdo compatível](#).

O API Gateway só compacta a carga de resposta quando as seguintes condições são atendidas:

- A solicitação de entrada tem o cabeçalho `Accept-Encoding` com uma codificação e formato de conteúdo compatíveis.

### Note

Se o cabeçalho não estiver configurado, o valor padrão será \* conforme definido na [RFC 7231](#). Nesse caso, o API Gateway não compacta a carga. Algum navegador ou cliente pode adicionar `Accept-Encoding` (por exemplo, `Accept-Encoding: gzip, deflate, br`) automaticamente para solicitações habilitadas para compactação. Isso pode ativar a compactação de carga útil no API Gateway. Sem uma especificação explícita dos valores de cabeçalho `Accept-Encoding` compatíveis, o API Gateway não compacta a carga.

- A definição de `minimumCompressionSize` na API habilita a compactação.
- A resposta de integração não tem um cabeçalho `Content-Encoding`.
- O tamanho da carga da resposta de integração, após a aplicação do modelo de mapeamento adequado, é maior ou igual ao valor especificado em `minimumCompressionSize`.

O API Gateway aplica qualquer modelo de mapeamento que estiver configurado para a resposta de integração antes de compactar a carga. Se a resposta de integração contiver um cabeçalho `Content-Encoding`, para o API Gateway, a carga da resposta de integração já estará compactada, e o processamento de compactação será ignorado.

Como exemplo, veja a API PetStore e a seguinte solicitação:

```
GET /pets
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
Accept: application/json
```

O backend responde à solicitação com uma carga JSON não compactada que é semelhante à seguinte:

```

200 OK

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

```

Para receber essa saída como uma carga compactada, o cliente da API pode enviar uma solicitação como esta:

```

GET /pets
Host: {petstore-api-id}.execute-api.{region}.amazonaws.com
Accept-Encoding:gzip

```

O cliente recebe a resposta com um cabeçalho Content-Encoding e a carga codificada por GZIP semelhante a esta:

```

200 OK
Content-Encoding:gzip
...

RP

J)V
:P^IeA*+++++(L XYZku0L0B7!9C#&++++Yaa^^X

```

Quando a carga de resposta é compactada, a cobrança pela transferência de dados leva em conta apenas o tamanho dos dados compactados.

# Distribuir as APIs REST para clientes no API Gateway

Esta seção fornece detalhes sobre como distribuir suas APIs do API Gateway para os clientes. A distribuição de sua API inclui a geração de SDKs que seus clientes podem baixar e integrar às suas aplicações de cliente, documentando a sua API para que saibam como chamá-la em suas aplicações de cliente e disponibilizando a API como parte das ofertas de produtos.

## Tópicos

- [Usar planos e chaves de API para APIs REST no APIs Gateway](#)
- [Documentação para APIs REST no API Gateway](#)
- [Gerar SDKs para APIs REST no API Gateway](#)
- [Venda suas APIs do API Gateway pelo AWS Marketplace](#)

## Usar planos e chaves de API para APIs REST no APIs Gateway

Depois que criar, testar e implantar suas APIs, você poderá usar planos de uso do API Gateway para disponibilizá-las como ofertas de produto aos seus clientes. É possível configurar planos de uso e chaves de API para permitir que os clientes acessem APIs selecionadas e inaugurem o controle de utilização de solicitações para essas APIs com base em limites e cotas definidos. Eles podem ser configurados na API ou no nível do método da API.

### O que são planos de uso e chaves de API?

Um plano de uso especifica quem poderá acessar um ou mais estágios e métodos da API implantada e, opcionalmente, define a taxa de solicitação de destino para iniciar o controle de utilização de solicitações. O plano usa chaves de API para identificar clientes de APIs e quem poderá acessar os estágios da API associada para cada chave.

Chaves de API são valores de strings alfanuméricas distribuídas para clientes de desenvolvedores de aplicativos para conceder acesso à API. Você pode usar chaves de API com [autorizadores do Lambda](#), [funções do IAM](#) ou o [Amazon Cognito](#) para controlar o acesso às suas APIs. O API Gateway pode gerar chaves de API em seu nome ou você pode importá-las para um [arquivo CSV](#). Você pode gerar uma chave de API no API Gateway ou importá-la para o API Gateway a partir de uma fonte externa. Para obter mais informações, consulte [the section called “Configurar chaves de API para a API REST no API Gateway”](#).

Uma chave de API tem um nome e um valor. (Os termos "chave de API" e "valor de chave de API" são frequentemente usados de forma intercambiável.) O nome não pode exceder 1.024 caracteres. O valor é uma string alfanumérica com tamanho entre 20 e 128 caracteres; por exemplo, `apikey1234abcdefgghij0123456789`.

 Important

Os valores de chaves de API devem ser exclusivos. Se você tentar criar duas chaves de API com nomes diferentes e o mesmo valor, o API Gateway as considera a mesma chave de API.

Uma chave de API pode ser associada a mais de um plano de uso. Um plano de uso pode ser associado a mais de um estágio. No entanto, uma determinada chave de API só pode ser associada a um único plano de uso para cada estágio de sua API.

Um limite de controle de utilização define o ponto de destino em que o controle de utilização de solicitações deverá ser iniciado. Isso pode ser configurado na API ou no nível do método da API.

Um limite de cota define o número máximo alvo de solicitações com uma determinada chave de API que poderão ser enviadas em um intervalo de tempo especificado. Você pode configurar métodos de API particulares para exigir a autorização da chave de API com base na configuração do plano de uso.

Limites de controle de utilização e cotas são aplicáveis a solicitações para chaves de API particulares que estão agregadas por todos os estágios de API em um plano de uso.

 Note

O controle de utilização e cotas do plano de uso não são limites rígidos e são aplicados de acordo com os melhores esforços. Em alguns casos, os clientes podem exceder as cotas definidas por você. Não dependa de cotas e controle de utilização de planos de uso para controlar custos ou bloquear o acesso a uma API. Considere usar o [AWS Budgets](#) para monitorar custos e o [AWS WAF](#) para gerenciar solicitações de APIs.

## Práticas Recomendadas para chaves de API e planos de uso

Veja a seguir sugestões de práticas recomendadas a serem seguidas ao usar chaves de API e planos de uso.



**⚠ Important**

- Não use chaves de API para autenticação ou autorização para controlar as APIs. Se você tiver várias APIs em um plano de uso, um usuário com uma chave de API válida para uma API nesse plano de uso poderá acessar todas as APIs desse plano. Em vez disso, para controlar o acesso à API, use um perfil do IAM, um [autorizador do Lambda](#) ou um [grupo de usuários do Amazon Cognito](#).
  - Use as chaves de API geradas pelo API Gateway. As chaves de API não devem incluir informações confidenciais; os clientes geralmente as transmitem em cabeçalhos que podem ser registrados em log.
- 
- Se você estiver usando um portal do desenvolvedor para publicar as APIs, observe que todas as APIs em determinado plano de uso podem ser assinadas por clientes, mesmo que você não as tenha tornado visíveis para os clientes.
  - Em alguns casos, os clientes podem exceder as cotas definidas por você. Não dependa de planos de uso para controlar custos. Considere usar [AWS Budgets](#) para monitorar custos e [AWS WAF](#) para gerenciar solicitações de APIs.
  - Depois que você adiciona uma chave de API a um plano de uso, a operação de atualização pode levar alguns minutos para ser concluída.

## Escolher uma origem de chaves de API no API Gateway

Quando você associa um plano de uso a uma API e habilita chaves de API em métodos de API, toda solicitação recebida para a API deve conter uma [chave de API](#). O API Gateway lê a chave e a compara com as chaves no plano de uso. Se houver correspondência, o API Gateway fará o controle de utilização das solicitações de acordo com o limite de solicitação e a cota do plano. Caso contrário, ele lançará uma exceção de `InvalidKeyParameter`. Como resultado, o autor da chamada recebe uma resposta 403 `Forbidden`.

A API do API Gateway pode receber chaves de API de duas origens:

### HEADER

Distribua chaves de API aos seus clientes e exija que eles repassem a chave de API como o cabeçalho `X-API-Key` de cada solicitação de entrada.

## AUTHORIZER

Um autorizador do Lambda pode retornar a chave de API como parte da resposta de autorização. Para obter mais informações sobre a resposta de autorização, consulte [the section called “Saída de um autorizador do Lambda para o API Gateway”](#).

### Note

Para ver as práticas recomendadas a serem consideradas, consulte [the section called “Práticas Recomendadas para chaves de API e planos de uso”](#).

O procedimento a seguir mostra como escolher uma origem de chave de API.

### Console de gerenciamento da AWS

Como escolher uma origem de chave de API para uma API

1. Inicie uma sessão no console do API Gateway.
2. Escolha uma API existente ou crie uma nova.
3. No painel de navegação principal, selecione Configurações da API.
4. Na seção Detalhes da API, escolha Editar.
5. Em Origem de chave de API, selecione Header ou Authorizer na lista suspensa.
6. Escolha Salvar alterações.

### AWS CLI

O comando [update-rest-api](#) indicado abaixo atualiza uma API para definir a origem da chave de API como AUTHORIZER:

```
aws apigateway update-rest-api --rest-api-id 1234123412 --patch-operations
op=replace,path=/apiKeySource,value=AUTHORIZER
```

Para que o cliente envie uma chave de API, defina o `value` como HEADER no comando anterior.

### REST API

Para escolher uma origem de chave de API para uma API usando a API REST do API Gateway, chame [restapi:update](#) da seguinte forma:

```

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 um autorizador retorne uma chave de API, defina `value` para `AUTHORIZER` na entrada `patchOperations` anterior.

## Formato de arquivo da chave de API do API Gateway

O API Gateway pode importar chaves de API de arquivos externos com um formato de valores separados por vírgula (CSV) e, depois, associar as chaves importadas a um ou mais planos de uso. O arquivo importado deve conter as colunas `Name` e `Key`. Os nomes de cabeçalhos de coluna não fazem distinção entre maiúsculas e minúsculas, e as colunas podem estar em qualquer ordem, como mostra o exemplo a seguir:

```

Key,name
apikey1234abcdefgghij0123456789,MyFirstApiKey

```

Um valor de `Key` deve ter entre 20 e 128 caracteres. Um valor `Name` não pode exceder 1.024 caracteres.

Um arquivo de chave de API também pode ter a coluna `Description`, `Enabled` ou `UsagePlanIds`, como mostra o exemplo a seguir:

```

Name,key,description,Enabled,usageplanIds

```

```
MyFirstApiKey,apikey1234abcdefg hij0123456789,An imported key,TRUE,c7y23b
```

Quando uma chave está associada a mais de um plano de uso, o valor de `UsagePlanIds` é uma string separada por vírgulas dos IDs do plano de uso e delimitada por um par de cotas duplas ou simples, como mostra o exemplo a seguir:

```
Enabled,Name,key,UsageplanIds
true,MyFirstApiKey,apikey1234abcdefg hij0123456789,"c7y23b,glvr sr"
```

Colunas não reconhecidas são permitidas, mas são ignoradas. O valor padrão é uma string vazia ou um valor booleano `true`.

A mesma chave de API pode ser importada várias vezes, com a versão mais recente substituindo a anterior. Duas chaves de API serão idênticas se tiverem o mesmo valor de `key`.

#### Note

Para ver as práticas recomendadas a serem consideradas, consulte [the section called “Práticas Recomendadas para chaves de API e planos de uso”](#).

## Configurar chaves de API para a API REST no API Gateway

Para configurar as chaves de API, faça o seguinte:

- Configure métodos de API para exigir uma chave de API.
- Crie ou importe uma chave de API para a API em uma região.

Antes de configurar chaves de API, é necessário ter criado a API e tê-la implantado em uma etapa. Depois que um valor de chave de API for atribuído, ele não poderá ser alterado.

Para obter instruções sobre como criar e implantar uma API usando o console do API Gateway, consulte [Desenvolver APIs REST no API Gateway](#) e [Implantar APIs REST no API Gateway](#), respectivamente.

Depois de criar uma chave de API, você deve associá-la a um plano de uso. Para obter mais informações, consulte [Configurar planos de uso para APIs REST no API Gateway](#).

 Note

Para ver as práticas recomendadas a serem consideradas, consulte [the section called “Práticas Recomendadas para chaves de API e planos de uso”](#).

## Tópicos

- [Exigir uma chave de API em um método](#)
- [Criar uma chave de API](#)
- [Importar chaves de API](#)

## Exigir uma chave de API em um método

O procedimento a seguir descreve como configurar um método de API para exigir uma chave de API.

## Console de gerenciamento da AWS

Para configurar um método de API para exigir uma chave de API

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST.
3. No painel de navegação principal do API Gateway, escolha Resources (Recursos).
4. Em Resources (Recursos), crie um novo método ou escolha um existente.
5. Na guia Solicitação de método, em Configurações de solicitação de método, escolha Editar.

The screenshot displays the Amazon API Gateway console interface. On the left, a navigation pane shows a resource tree with the path `/pets` selected under the `GET` method. The main content area is titled 'Method request settings' and contains several configuration sections:

- Method request settings:** Includes fields for 'Authorization' (set to 'NONE'), 'Request validator' (set to 'None'), 'API key required' (set to 'False'), and 'SDK operation name' (set to 'Generated based on method and path'). An 'Edit' button is visible in the top right corner of this section.
- Request paths (0):** A section indicating that no request paths are currently defined.
- URL query string parameters (2):** A table listing two parameters: 'page' and 'type'. Both are marked as 'Required: False' and 'Caching: Inactive'.
- HTTP request headers (2):** A table listing one header: 'header1', marked as 'Required: False' and 'Caching: Inactive'.

6. Selecione a Chave de API obrigatória.
7. Escolha Salvar.
8. Implante ou reimplante a API para que o requisito entre em vigor.

Se a opção Chave de API obrigatória estiver definida como `false` e você não executar as etapas anteriores, nenhuma chave de API associada a um estágio de API será usada para o método.

## AWS CLI

O comando [put-method](#) indicado abaixo cria um método PUT que exige uma chave de API:

```
aws apigateway put-method \
 --rest-api-id 1234123412 \
 --resource-id a1b2c3 \
 --http-method PUT \
 --authorization-type "NONE" \
```

```
--api-key-required
```

O comando [update-method](#) indicado abaixo atualiza um método existente para exigir uma chave 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 uma chave de API em um método, siga um destes procedimentos:

- Chame [method:put](#) para criar um método. Defina `apiKeyRequired` como `true` na carga da solicitação.
- Chame [method:update](#) para definir `apiKeyRequired` como `true`.

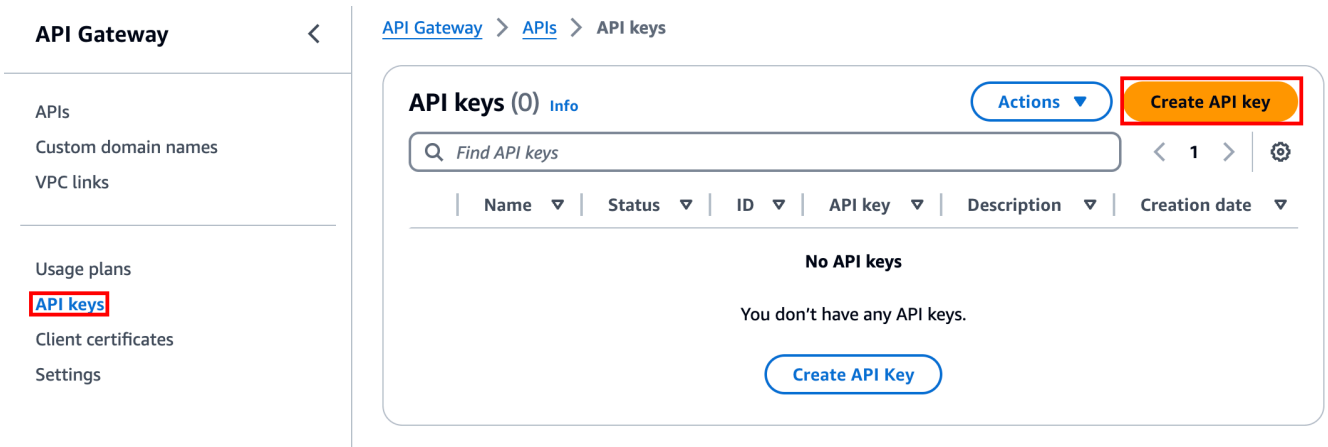
## Criar uma chave de API

O procedimento a seguir mostra como criar uma chave de API. Se você quiser importar a chave de API, ignore esta etapa.

### Console de gerenciamento da AWS

Para criar uma chave de API

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST.
3. No painel de navegação principal do API Gateway, escolha Chaves de API.
4. Escolha Criar chave de API.



5. Em Nome, insira um nome.
6. (Opcional) Em Description (Descrição), insira uma descrição.
7. Em Chave de API, escolha Gerar automaticamente para que o API Gateway gere o valor da chave ou escolha Personalizado para criar seu próprio valor de chave.
8. Escolha Salvar.

## AWS CLI

O comando [create-api-key](#) indicado abaixo cria uma chave de API:

```
aws apigateway create-api-key \
 --name 'Dev API key' \
 --description 'API key for Devs' \
 --enabled
```

## REST API

Chame [apikey:create](#) para criar uma chave de API.

## Importar chaves de API

O procedimento a seguir descreve como importar chaves de API. Se você já tiver criado uma chave de API, ignore esta etapa.



## Console de gerenciamento da AWS

Para importar chaves de API

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST.
3. No painel de navegação principal, selecione Chaves de API.
4. Selecione o menu suspenso Ações e Importar chaves de API.
5. Para carregar um arquivo de chave separado por vírgula, selecione Escolher arquivo. Você também pode inserir as chaves no editor de texto. Para obter informações sobre o formato do arquivo, consulte [the section called “Formato de arquivo da chave de API do API Gateway”](#).
6. Escolha Falhar nos avisos para interromper a importação em caso de erro ou escolha Ignorar avisos para continuar a importar entradas de chave válidas quando houver um erro.
7. Escolha Importar para importar as chaves de API.

## AWS CLI

O comando [import-api-keys](#) indicado abaixo importa uma chave de API:

```
aws apigateway import-api-key \
 a--body fileb://keys.csv \
 --format csv
```

## REST API

Chame [apikey:import](#) para importar uma chave de API de um arquivo. Para saber o formato de arquivo, consulte [Formato de arquivo da chave de API do API Gateway](#).

Não é possível alterar o valor da nova chave de API. Depois de criar a API, configure um plano de uso. Para obter mais informações, consulte [Configurar planos de uso para APIs REST no API Gateway](#).

## Configurar planos de uso para APIs REST no API Gateway

Antes de criar um plano de uso, você deve configurar as chaves de API. Para obter mais informações, consulte [Configurar chaves de API para a API REST no API Gateway](#).

## Tópicos

- [Migrar a API para planos de uso padrão \(se necessário\)](#)
- [Criar um plano de uso](#)
- [Adicionar um estágio de API a um plano de uso](#)
- [Adicionar uma chave de API a um plano de uso](#)

### Migrar a API para planos de uso padrão (se necessário)

Se você começou a usar o API Gateway depois que o recurso de planos de uso foi lançado em 11 de agosto de 2016, terá planos de uso habilitados automaticamente em todas as regiões compatíveis.

Se começou a usar o API Gateway antes dessa data, poderá ser necessário migrar para os planos de uso padrão. Você receberá a opção Enable Usage Plans (Habilitar planos de uso) antes de usar os planos de uso pela primeira vez na região selecionada. Quando você habilita essa opção, você tem planos de uso padrão criados para cada estágio de API exclusivo que está associado a chaves de API existentes. No plano de uso padrão, nenhum limite de controle ou limite de cota é definido inicialmente, e as associações entre chaves de API e estágios de API são copiadas para os planos de uso. A API tem o mesmo comportamento de antes. Contudo, você deve usar a propriedade de [UsagePlan](#) `apiStages` para associar valores especificados de estágios de API (`apiId` e `stage`) com as chaves de API incluídas (via [UsagePlanKey](#)), em vez de usar a propriedade `stageKeys` da [ApiKey](#).

Para verificar se você já migrou para os planos de uso padrão, use o comando [get-account](#) da CLI. Na saída do comando, a lista `features` inclui uma entrada de `"UsagePlans"` quando os planos de uso estão habilitados.

Você também pode migrar as APIs para os planos de uso padrão usando a AWS CLI da seguinte maneira:

Para migrar para os planos de uso padrão usando a AWS CLI

1. Chame este comando da CLI: [update-account](#).
2. No parâmetro `cli-input-json`, use o seguinte JSON:

```
[
 {
 "op": "add",
 "path": "/features",
```

```
 "value": "UsagePlans"
 }
]
```

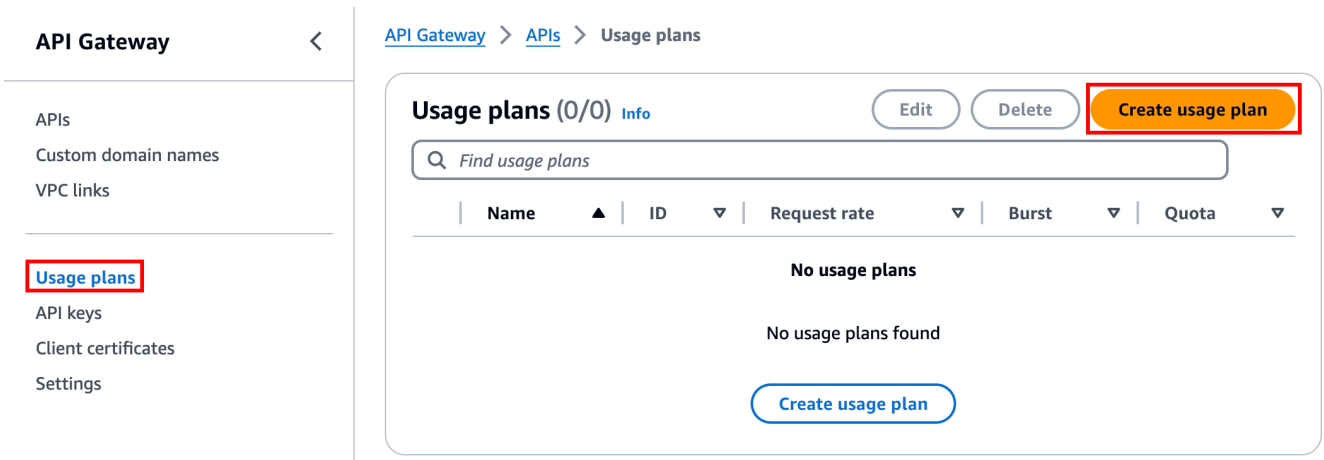
## Criar um plano de uso

O procedimento a seguir descreve como criar um plano de uso.

### Console de gerenciamento da AWS

Para criar um plano de uso

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal do API Gateway, escolha Planos de uso e Criar plano de uso.



3. Em Nome, insira um nome.
4. (Opcional) Em Description (Descrição), insira uma descrição.
5. Por padrão, os planos de uso permitem o controle de utilização. Insira uma Taxa e um Pico para seu plano de uso. Escolha Controle de utilização para desativar o controle de utilização.
6. Por padrão, os planos de uso permitem uma cota por um período. Em Solicitações, insira o número total de solicitações que um usuário pode fazer no período do plano de uso. Escolha Cota para desativar a cota.
7. Escolha Criar plano de uso.

## AWS CLI

O comando [create-usage-plan](#) indicado abaixo criar um plano de uso que é redefinido no início do mês:

```
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

Chame [usageplan:create](#) para criar um plano de uso.

Adicionar um estágio de API a um plano de uso

O procedimento a seguir descreve como adicionar um estágio a um plano de uso.

### Console de gerenciamento da AWS

Como adicionar um estágio a um plano de uso

1. Selecione o plano de uso.
2. Na guia Estágios associados, escolha Adicionar estágio.

[API Gateway](#) > [APIs](#) > [Usage plans](#) > MyUsagePlan

## MyUsagePlan

Actions ▼

Export usage data

### Usage plan details

**Usage plan ID**  
abc123**Description**  
My new usage plan**AWS Marketplace product code**  
-**Rate**  
100 requests per second**Burst**  
20 requests**Quota**  
10 requests per month**Associated stages**

Associated API keys

Tags

### Associated stages (0) [Info](#)

Edit

Remove

**Add stage**

API ▼

Stage ▼

Method throttling

**No stages**

You don't have any stages.

**Add API stage**

3. Em API, selecione uma API.
4. Em Estágio, selecione um estágio.
5. (Opcional) Para ativar o controle de utilização no nível do método, faça o seguinte:
  - a. Escolha Controle de utilização no nível do método e Adicionar método.
  - b. Em Recurso, selecione um recurso da API.
  - c. Em Método, selecione um método da API.
  - d. Insira uma Taxa e um Pico para seu plano de uso.
6. Escolha Adicionar ao plano de uso.

## AWS CLI

O comando [update-usage-plan](#) indicado abaixo adiciona o estágio Prod de uma API a um plano de uso:

```
aws apigateway update-usage-plan \
 --usage-plan-id abc123 \
 --patch-operations op="add",path="/apiStages",value="a1b1c2:Prod"
```

## REST API

Chame [usageplan:update](#) para atualizar um plano de uso.

### Adicionar uma chave de API a um plano de uso

O procedimento a seguir mostra como adicionar uma chave de API a um plano de uso.

### Console de gerenciamento da AWS

Como adicionar uma chave a um plano de uso

1. Na guia Chaves de API associadas, escolha Adicionar chave de API.

## MyUsagePlan

Actions ▼

Export usage data

### Usage plan details

**Usage plan ID**

abc123

**Description**

My new usage plan

**AWS Marketplace product code**

-

**Rate**

100 requests per second

**Burst**

20 requests

**Quota**

10 requests per month

Associated stages

Associated API keys

Tags

### API keys (0) [Info](#)

Actions ▼

Add API key

&lt; 1 &gt; ⚙

Name ▼

Status ▼

ID ▼

API key ▼

Requests remaining this month ▼

No API keys.

This usage plan has no API keys.

Add API key

2.
  - a. Para associar uma chave existente ao plano de uso, selecione Adicionar chave existente e escolha a respectiva chave no menu suspenso.
  - b. Para criar uma chave de API, selecione Criar e adicionar nova chave e crie uma chave. Para obter mais informações sobre como criar uma chave, consulte [Criar uma chave de API](#).
3. Escolha Adicionar chave de API.

## AWS CLI

O comando [create-usage-plan-key](#) indicado abaixo associa uma chave de API existente a um plano de uso:

```
aws apigateway create-usage-plan-key \
 --usage-plan-id a1b2c3 \
 --key-type "API_KEY" \
 --key-id aaa111bbb
```

## REST API

Chame [usageplankey:create](#) para associar uma chave de API existente a um plano de uso.

Você também pode associar diretamente as chaves de API a um plano de uso ao importá-las. Chame [apikey:import](#) para adicionar uma ou mais chaves de API diretamente ao plano de uso especificado. A carga da solicitação deve conter valores de chaves de API, o identificador de plano de uso associado, os sinalizadores booleanos para indicar que as chaves estão habilitadas para o plano de uso e, possivelmente, os nomes e as descrições das chaves de API.

O exemplo a seguir da solicitação `apikey:import` adiciona três chaves de API (identificadas por `key`, `name` e `description`) a um plano de um uso (identificado por `usageplanIds`):

```
POST /apikeys?mode=import&format=csv&failonwarnings=false HTTP/1.1
Host: apigateway.us-east-1.amazonaws.com
Content-Type: text/csv
Authorization: ...

key,name, description, enabled, usageplanIds
abcdef1234ghijklmnop8901234567, importedKey_1, firstone, tRuE, n371pt
abcdef1234ghijklmnop0123456789, importedKey_2, secondone, TRUE, n371pt
abcdef1234ghijklmnop9012345678, importedKey_3, , true, n371pt
```

Como resultado, três recursos `UsagePlanKey` são criados e adicionados ao `UsagePlan`.

Você também pode adicionar chaves de API a mais de um plano de uso dessa maneira. Para fazer isso, altere cada valor de coluna `usageplanIds` para uma string separada por vírgulas que contenha os identificadores do plano de uso selecionado e que esteja dentro de um par de aspas ("`n371pt,m282qs`" ou '`n371pt,m282qs`').



**Note**

Uma chave de API pode ser associada a mais de um plano de uso. Um plano de uso pode ser associado a mais de um estágio. No entanto, uma determinada chave de API só pode ser associada a um único plano de uso para cada estágio de sua API.

## Manter um plano de uso para APIs REST no API Gateway

A manutenção de um plano de uso requer o monitoramento das cotas usadas e restantes durante determinado período e, se necessário, a extensão das cotas restantes de acordo com uma quantidade especificada. Os procedimentos a seguir descrevem como monitorar cotas.

### Console de gerenciamento da AWS

Para monitorar as cotas usadas e restantes

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal do API Gateway, escolha Planos de uso.
3. Selecione um plano de uso.
4. Escolha a guia Chaves de API associadas para ver o número de solicitações restantes no período de cada chave.
5. (Opcional) Escolha Exportar dados de uso e escolha uma data para De e uma data para Até. Depois, escolha JSON ou CSV para o formato de dados exportados e selecione Exportar.

O exemplo a seguir mostra um arquivo exportado.

```
{
 "px1KW6...qBaz0JH": [
 [
 0,
 5000
],
 [
 0,
 5000
],
 [
 0,
 10
]
]
}
```

```
]
]
}
```

Os dados de uso no exemplo mostram os dados de uso diários para um cliente de API, conforme identificado pela chave de API (px1KW6 . . . qBaz0JH), entre 1º de agosto de 2016 e 3 de agosto de 2016. Cada dado de uso diário mostra as cotas usadas e restantes. Nesse exemplo, o assinante não utilizou as cotas alocadas ainda, e o proprietário ou o administrador da API reduziu a cota restante de 5000 para 10 no terceiro dia.

Os procedimentos a seguir descrevem como modificar cotas.

Para estender as cotas restantes

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal do API Gateway, escolha Planos de uso.
3. Selecione um plano de uso.
4. Escolha a guia Chaves de API associadas para ver o número de solicitações restantes no período de cada chave.
5. Selecione uma chave de API e escolha Conceder extensão de uso.
6. Insira um número para a cota de Solicitações restantes. Você pode aumentar as solicitações de renomeação ou diminuir as solicitações restantes durante o período do plano de uso.
7. Escolha Atualizar cota.

## AWS CLI

Os exemplos de código [update-usage-plan](#) a seguir adicionam, removem ou modificam as configurações de controle de utilização em nível de método em um plano de uso.

### Note

Altere o `us-east-1` para o valor de região apropriado para sua API.

Para adicionar ou substituir um limite de taxa para limitação de uso de um recurso e método específico:

```
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 adicionar ou substituir um limite de intermitência para limitação de uso de um recurso e método específico:

```
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 remover as configurações de limitação de uso de um recurso e método específico:

```
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 remover todas as configurações de limitação de uso em nível de método para uma API:

```
aws apigateway --region us-east-1 update-usage-plan --usage-plan-id planId --patch-operations op="remove",path="/apiStages/apiId:stage/throttle ",value=""
```

Veja um exemplo que usa a API de exemplo PetStore:

```
aws apigateway --region us-east-1 update-usage-plan --usage-plan-id planId --patch-operations op="replace",path="/apiStages/apiId:stage/throttle",value='{"\n/pets/GET\n":{"rateLimit\n":1.0,\n"burstLimit\n":1},\n"/GET\n":{"rateLimit\n":1.0,\n"burstLimit\n":1}}'
```

## REST API

Chame [usageplan:update](#) para manter um plano de uso.

## Criar e configurar chaves de API e planos de uso com o CloudFormation

É possível usar o CloudFormation para exigir chaves de API em métodos de API e criar um plano de uso para uma API. O modelo do CloudFormation de exemplo faz o seguinte:

- Cria uma API do API Gateway com os métodos GET e POST.

- Exige uma chave de API para os métodos GET e POST. Essa API recebe chaves do cabeçalho X-API-KEY de cada solicitação recebida.
- Cria uma chave de API.
- Cria um plano de uso para especificar uma cota mensal de 1.000 solicitações por mês, um limite de taxa de controle de utilização de 100 solicitações por segundo e um limite de intermitência de controle de utilização de 200 solicitações por segundo.
- Especifica um limite de taxa de controle de utilização no nível do método de 50 solicitações por segundo e um limite de intermitência de controle de utilização no nível do método de 100 solicitações por segundo para o método GET.
- Associa o estágio da API e a chave de API ao plano 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}'

```

## Configurar um método para usar chaves de API com uma definição OpenAPI

Você pode usar uma definição OpenAPI para exigir chaves de API em um método.

Para cada método, crie um objeto de requisito de segurança a fim de exigir uma chave de API para invocar esse método. Em seguida, defina `api_key` na configuração de segurança. Depois de criar a API, adicione o estágio da nova API ao plano de uso.

O exemplo a seguir cria uma API e exige uma chave de API para os métodos POST e 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"
 }
 }
 }
}
```



## Testar planos de uso para APIs REST no API Gateway

Como exemplo, vamos usar a API PetStore, que foi criada em [Tutorial: Criar uma API REST importando um exemplo](#). Suponha que essa API esteja configurada para usar uma chave de API de `Hiorr45VR...c4GJc`. As etapas a seguir descrevem como testar um plano de uso.

Para testar seu plano de uso

- Faça uma solicitação GET no recurso Pets (`/pets`), com os parâmetros de consulta `?type=...&page=...`, da API (por exemplo, `xbvxlpijch`) em um plano 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

É necessário enviar essa solicitação ao componente `execute-api` do API Gateway e fornecer a chave de API necessária (por exemplo, `Hiorr45VR...c4GJc`) no cabeçalho `x-api-key` exigido.

A resposta bem-sucedida retorna um código de status `200 OK` e uma carga que contém os resultados solicitados do backend. Se você se esquecer de definir o cabeçalho `x-api-key` ou se defini-lo com uma chave incorreta, você recebe uma resposta `403 Forbidden`. No entanto, se você não configurou o método para exigir uma chave de API, provavelmente obterá uma resposta `200 OK` independentemente ou não de definir o cabeçalho `x-api-key` corretamente e os limites de cota e controle de fluxo do plano de uso serão ignorados.

Ocasionalmente, quando ocorrer um erro interno em que o API Gateway fica incapaz de impor limites de controle de utilização ou cotas para a solicitação, o API Gateway atenderá a essa solicitação sem aplicar esses limites ou cotas, conforme especificado no plano de uso. No entanto, registrará uma mensagem de erro de `Usage Plan check failed due to an internal error` no CloudWatch. Você pode ignorar esses erros ocasionais.

## Chamar um método usando uma chave de API

Dependendo do tipo escolhido de origem da chave de API, use um dos seguintes procedimentos para usar chaves de API originadas de cabeçalho ou chaves de API retornadas por um autorizador em um método de invocação:

Para usar chaves de API originadas de cabeçalho:

1. Crie uma API com os métodos de API desejados e implante a API em um estágio.
2. Crie um novo plano de uso ou escolha um novo. Adicione o estágio de API implantado ao plano de uso. Anexe uma chave de API ao plano de uso ou escolha uma chave de API existente no plano. Observe o valor da chave de API escolhido.
3. Configure métodos de API para exigir uma chave de API.
4. Reimplante a API para o mesmo estágio. Se você implantar a API para um novo estágio, certifique-se de atualizar o plano de uso a fim de anexar o novo estágio de API.
5. Chame a API usando a chave de API. O exemplo de comando curl a seguir invoca o método GET no recurso getUsers do estágio prod de uma API usando uma chave de API.

```
curl -H "X-API-Key: abcd1234" 'https://b123abcde4.execute-api.us-west-2.amazonaws.com/prod/getUsers'
```

Agora o cliente pode chamar métodos de API enquanto fornece o cabeçalho x-api-key com a chave de API escolhida como o valor do cabeçalho. Uma chamada pode ser semelhante ao seguinte:

Para usar chaves de API originadas de um autorizador:

1. Crie uma API com os métodos de API desejados e implante a API em um estágio.
2. Crie um novo plano de uso ou escolha um novo. Adicione o estágio de API implantado ao plano de uso. Anexe uma chave de API ao plano de uso ou escolha uma chave de API existente no plano. Observe o valor da chave de API escolhido.
3. Crie um autorizador do Lambda com base em token. Inclua `usageIdentifierKey: {api-key}` como uma propriedade em nível de raiz da resposta de autorização. Para encontrar instruções sobre como criar um autorizador baseado em token, consulte [the section called “Exemplo de função do Lambda do autorizador TOKEN”](#).

4. Configure métodos de API para exigir uma chave de API e habilite o autorizador do Lambda nos métodos.
5. Reimplante a API para o mesmo estágio. Se você implantar a API para um novo estágio, certifique-se de atualizar o plano de uso a fim de anexar o novo estágio de API.

Agora, o cliente pode chamar os métodos de API que exigem chave sem fornecer explicitamente qualquer chave de API. A chave de API retornada pelo autorizador é usada automaticamente.

## Documentação para APIs REST no API Gateway

Para ajudar os clientes a compreender e usar sua API, você deve documentá-la. Para ajudar você a documentar sua API, o API Gateway permite adicionar e atualizar o conteúdo da ajuda para entidades de API individuais como parte integrante do seu processo de desenvolvimento de APIs. O API Gateway armazena o conteúdo de origem e permite arquivar diferentes versões da documentação. É possível associar uma versão de documentação a um estágio da API, exportar um snapshot da documentação específico de um estágio para um arquivo do OpenAPI externo e distribuir esse arquivo como uma publicação da documentação.

Para documentar uma API, você pode chamar a [API REST do API Gateway](#), usar um dos [AWS SDKs](#) ou a [AWS CLI](#) para o API Gateway ou usar o console do API Gateway. Além disso, é possível importar ou exportar as partes da documentação definidas em um arquivo OpenAPI externo.

Para compartilhar a documentação da API com os desenvolvedores, você pode usar um portal para desenvolvedores. Por exemplo, consulte [Integrating ReadMe with API Gateway to Keep Your Developer Hub Up to Date](#) ou [How to Streamline API Development on Amazon API Gateway Using SmartBear's SwaggerHub](#) no blog da Rede de Parceiros da AWS (APN).

### Tópicos

- [Representação da documentação da API no API Gateway](#)
- [Documentar uma API usando o console do API Gateway](#)
- [Publicar a documentação da API usando o console do API Gateway](#)
- [Documentar uma API usando a API REST do API Gateway](#)
- [Publicar a documentação da API usando a API REST do API Gateway](#)
- [Importar a documentação da API](#)
- [Controlar o acesso à documentação da API no API Gateway](#)

## Representação da documentação da API no API Gateway

A documentação de API do API Gateway consiste em partes de documentação individuais associadas a entidades de API específicas que incluem API, recurso, método, solicitação, resposta, parâmetros de mensagem (ou seja, caminho, consulta, cabeçalho), bem como autorizadores e modelos.

No API Gateway, uma parte de documentação é representada por um recurso [DocumentationPart](#). A documentação da API como um todo é representada pela coleção [DocumentationParts](#).

Documentar uma API envolve criar instâncias de `DocumentationPart`, adicioná-las à coleção `DocumentationParts` e manter versões das partes de documentação à medida que a sua API evolui.

### Tópicos

- [Partes da documentação](#)
- [Versões da documentação](#)

### Partes da documentação

Um recurso [DocumentationPart](#) é um objeto JSON que armazena o conteúdo da documentação aplicável a uma entidade da API individual. Isso inclui qualquer conteúdo UTF-8 e todos os principais idiomas de localização para sua documentação. Seu campo `properties` contém o conteúdo da documentação como um mapa de pares de chave/valor. Sua propriedade `location` identifica a entidade de API associada.

A forma de um mapa de conteúdo é determinada por você, o desenvolvedor da API. O valor de um par de chave/valor pode ser uma string, número, booleano, objeto ou matriz. A forma do objeto `location` depende do tipo de entidade alvo.

O recurso `DocumentationPart` oferece suporte para herança de conteúdo: o conteúdo da documentação de uma entidade de API é aplicável aos filhos dessa entidade de API. Para obter mais informações sobre a definição de entidades filho e herança de conteúdo, consulte [Herdar conteúdo de uma entidade de API com especificação mais geral](#).

### Localização de uma parte de documentação

A propriedade `location` de uma instância de [DocumentationPart](#) identifica uma entidade de API à qual o conteúdo associado se aplica. A entidade de API pode ser um recurso de API REST do API Gateway, como [RestApi](#), [Resource](#), [Method](#), [MethodResponse](#), [Authorizer](#) ou [Model](#). Ela também

pode ser um parâmetro de mensagem, como um parâmetro de caminho de URL, um parâmetro de string de consulta, um parâmetro de cabeçalho de solicitação ou resposta, um corpo de solicitação ou resposta ou um código de status de resposta.

Para especificar uma entidade da API, defina o atributo [type](#) do objeto `location` como um dos seguintes: `API`, `AUTHORIZER`, `MODEL`, `RESOURCE`, `METHOD`, `PATH_PARAMETER`, `QUERY_PARAMETER`, `REQUEST_HEADER`, `REQUEST_BODY`, `RESPONSE`, `RESPONSE_HEADER` ou `RESPONSE_BODY`.

Dependendo do `type` de uma entidade da API, você pode especificar outros atributos de `location`, incluindo [method](#), [name](#), [path](#) e [statusCode](#). Nem todos esses atributos são válidos para uma determinada entidade de API. Por exemplo, `type`, `path`, `name` e `statusCode` são atributos válidos da entidade `RESPONSE`; apenas `type` e `path` são atributos de localização válidos da entidade `RESOURCE`. É um erro incluir um campo inválido no `location` de um `DocumentationPart` para uma determinada entidade de API.

Nem todos os campos válidos de `location` são necessários. Por exemplo, `type` é o campo `location` tanto válido quanto obrigatório de todas as entidades de API. No entanto, `method`, `path` e `statusCode` são atributos válidos, mas não obrigatórios, para a entidade `RESPONSE`. Quando não explicitamente especificado, um campo `location` válido assume seu valor padrão. O valor de `path` padrão é `/`, ou seja, o recurso raiz de uma API. O valor padrão de `method`, ou `statusCode`, é `*`, significando qualquer valor de método ou código de status, respectivamente.

### Conteúdo de uma parte da documentação

O valor de `properties` é codificado como uma string JSON. O valor de `properties` contém qualquer informação que você escolher para atender às suas necessidades de documentação. Por exemplo, o seguinte é um mapa de conteúdo 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."
}
```

Embora o API Gateway aceite qualquer string JSON válida como o mapa de conteúdo, os atributos de conteúdo são tratados como duas categorias: aqueles que podem ser reconhecidos pelo OpenAPI e aqueles que não podem. No exemplo anterior, `info`, `description` e `x-custom-info`

são reconhecidos pelo OpenAPI como um objeto, uma propriedade ou uma extensão padrão do OpenAPI. Em contraste, `my-info` não é compatível com a especificação OpenAPI. O API Gateway propaga atributos de conteúdo compatíveis com OpenAPI para as definições de entidades de API das instâncias `DocumentationPart` associadas. O API Gateway não propaga os atributos de conteúdo não compatíveis nas definições de entidades de API.

Como outro exemplo, aqui está uma `DocumentationPart` direcionada para uma entidade `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...",
 }
}
```

Aqui, tanto `type` quanto `path` são campos válidos para identificar o destino do tipo `RESOURCE`. Para o recurso de raiz (`/`), você pode omitir o campo `path`.

```
{
 "location" : {
 "type" : "RESOURCE"
 },
 "properties" : {
 "description" : "The root resource with the default path specification."
 }
}
```

Isso equivale à seguinte instância de `DocumentationPart`:

```
{
 "location" : {
 "type" : "RESOURCE",
 "path": "/"
 },
 "properties" : {
 "description" : "The root resource with an explicit path specification"
 }
}
```

}

## Herdar conteúdo de uma entidade de API com especificações mais gerais

O valor padrão de um campo `location` opcional fornece uma descrição padronizada de uma entidade de API. Usando o valor padrão no objeto `location`, é possível adicionar uma descrição geral no mapa `properties` a uma instância de `DocumentationPart` com esse tipo de padrão de `location`. O API Gateway extrai os atributos da documentação do OpenAPI aplicáveis de `DocumentationPart` da entidade de API genérica e os injeta em uma entidade de API específica com os campos `location` correspondentes ao padrão `location` geral ou correspondentes ao valor exato, a menos que a entidade específica já tenha uma instância de `DocumentationPart` associada a ela. Esse comportamento também é conhecido como herança de conteúdo de uma entidade de API de especificações mais gerais.

A herança de conteúdo não se aplica a determinados tipos de entidade de API. Consulte a tabela a seguir para obter detalhes.

Quando uma entidade de API corresponder ao padrão de localização de `DocumentationPart`, ela herdará a parte de documentação com os campos de localização de maior precedência e especificidade. A ordem de precedência é `path` > `statusCode`. Para correspondência com o campo `path`, o API Gateway escolhe a entidade com o valor do caminho mais específico. A tabela a seguir mostra isso com alguns exemplos.

| Caso | path  | stat<br>e | name | Observ<br>es                                                                               |
|------|-------|-----------|------|--------------------------------------------------------------------------------------------|
| 1    | /pets | *         | id   | A<br>documenta<br>ção<br>associada<br>a<br>esse<br>padrão<br>de<br>localizaç<br>ão<br>será |

| Caso | path | stat | name | Observ    |  |
|------|------|------|------|-----------|--|
|      |      | e    |      | es        |  |
|      |      |      |      | herdada   |  |
|      |      |      |      | por       |  |
|      |      |      |      | entidades |  |
|      |      |      |      | que       |  |
|      |      |      |      | correspon |  |
|      |      |      |      | derem     |  |
|      |      |      |      | ao        |  |
|      |      |      |      | padrão    |  |
|      |      |      |      | de        |  |
|      |      |      |      | localizaç |  |
|      |      |      |      | ão.       |  |



| Caso | path  | stat<br>e | name | Observ<br>es                                                                                                                                                                                                                                                                           |  |
|------|-------|-----------|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| 2    | /pets | 200       | id   | A<br>documenta<br>ção<br>associada<br>a<br>esse<br>padrão<br>de<br>localizaç<br>ão<br>será<br>herdada<br>por<br>entidades<br>que<br>correspon<br>derem<br>ao<br>padrão<br>de<br>localizaç<br>ão<br>quando<br>tanto<br>o<br>Caso<br>1<br>quanto<br>o<br>Caso<br>2<br>forem<br>correspon |  |

| Caso | path | stat | name | Observ                                                                                      |  |
|------|------|------|------|---------------------------------------------------------------------------------------------|--|
|      |      | e    |      | es                                                                                          |  |
|      |      |      |      | didados,<br>pois<br>o<br>Caso<br>2 é<br>mais<br>específic<br>o do<br>que<br>o<br>Caso<br>1. |  |

| Caso | path            | stat<br>e | name | Observ<br>es                                                                                                                                                                                                                                                                                 |  |
|------|-----------------|-----------|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| 3    | /pets/<br>petId | *         | id   | A<br>documenta<br>ção<br>associada<br>a<br>esse<br>padrão<br>de<br>localizaç<br>ão<br>será<br>herdada<br>por<br>entidades<br>que<br>correspon<br>derem<br>ao<br>padrão<br>de<br>localizaç<br>ão<br>quando<br>os<br>Casos<br>1, 2<br>e 3<br>forem<br>correspon<br>didos,<br>pois<br>o<br>Caso |  |

| Caso | path | stat<br>e | name | Observ<br>es                                                                                                                   |  |
|------|------|-----------|------|--------------------------------------------------------------------------------------------------------------------------------|--|
|      |      |           |      | 3<br>tem<br>precedênc<br>ia<br>maior<br>do<br>que<br>o<br>Caso<br>2 e é<br>mais<br>específic<br>o do<br>que<br>o<br>Caso<br>1. |  |

Veja a seguir outro exemplo para contrastar uma instância mais genérica de `DocumentationPart` com uma mais específica. A seguinte mensagem de erro geral `"Invalid request error"` será injetada nas definições do OpenAPI das respostas de erro 400, a menos que ela seja substituída.

```
{
 "location" : {
 "type" : "RESPONSE",
 "statusCode": "400"
 },
 "properties" : {
 "description" : "Invalid request error."
 }
}
```

Com a seguinte substituição, as respostas 400 a qualquer método no recurso `/pets` têm uma descrição de `"Invalid petId specified"` em vez disso.

```
{
 "location" : {
 "type" : "RESPONSE",
 "path": "/pets",
 "statusCode": "400"
 },
 "properties" : "{
 "description" : "Invalid petId specified."
 }"
}
```

### Campos de localização válidos de **DocumentationPart**

A seguinte tabela mostra os campos válidos e necessários, bem como valores padrão aplicáveis de um recurso [DocumentationPart](#) que está associado a um determinado tipo de entidade de API.

| Entidade de API                 | Campos de localização válidos                                                                               | Campos de localização obrigatórios | Valores de campo padrão                | Conteúdo herdável        |
|---------------------------------|-------------------------------------------------------------------------------------------------------------|------------------------------------|----------------------------------------|--------------------------|
| <a href="#">solicitações de</a> | <pre>{   "location": {     "type": "API"   },   ... }</pre>                                                 | type                               | N/D                                    | Não                      |
| <a href="#">Recurso</a>         | <pre>{   "location": {     "type": "RESOURCE"   },   "path":     "<i>resource_path</i> "   },   ... }</pre> | type                               | O valor padrão de path é /.            | Não                      |
| <a href="#">Method (Método)</a> | <pre>{   "location": {</pre>                                                                                | type                               | Os valores padrão de path e method são | Sim, correspondendo path |

| Entidade de API       | Campos de localização válidos                                                                                                                                                                                   | Campos de localização obrigatórios | Valores de campo padrão                                        | Conteúdo herdável                                                                |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------|----------------------------------------------------------------|----------------------------------------------------------------------------------|
|                       | <pre>       "type":         "METHOD",       "path":         "<i>resource_path</i> ",       "method":         "<i>http_verb</i> "     },     ...   } </pre>                                                      |                                    | / e *, respectivamente.                                        | por prefixo e correspondendo method de quaisquer valores.                        |
| Parâmetro de consulta | <pre> {   "location": {     "type": "QUERY_PARAMETER",     "path":       "<i>resource_path</i> ",     "method":       "<i>HTTP_verb</i> ",     "name":       "<i>query_parameter_name</i> "   },   ... } </pre> | type                               | Os valores padrão de path e method são / e *, respectivamente. | Sim, correspondendo path por prefixo e correspondendo method por valores exatos. |
| Corpo da solicitação  | <pre> {   "location": {     "type": "REQUEST_BODY",     "path":       "<i>resource_path</i> ",     "method":       "<i>http_verb</i> "   },   ... } </pre>                                                      | type                               | Os valores padrão de path e method são / e *, respectivamente. | Sim, correspondendo path por prefixo e correspondendo method por valores exatos. |

| Entidade de API                       | Campos de localização válidos                                                                                                                                                                                                | Campos de localização obrigatórios | Valores de campo padrão                                        | Conteúdo herdável                                                                |
|---------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------|----------------------------------------------------------------|----------------------------------------------------------------------------------|
| Parâmetro do cabeçalho da solicitação | <pre>{   "location": {     "type": "REQUEST_HEADER",     "path":       "<i>resource_path</i> ",     "method":       "<i>HTTP_verb</i> ",     "name":       "<i>header_name</i> "   },   ... }</pre>                          | type, name                         | Os valores padrão de path e method são / e *, respectivamente. | Sim, correspondendo path por prefixo e correspondendo method por valores exatos. |
| Parâmetro do caminho da solicitação   | <pre>{   "location": {     "type": "PATH_PARAMETER",     "path":       "<i>resource/{path_parameter_name}</i> ",     "method":       "<i>HTTP_verb</i> ",     "name":       "<i>path_parameter_name</i> "   },   ... }</pre> | type, name                         | Os valores padrão de path e method são / e *, respectivamente. | Sim, correspondendo path por prefixo e correspondendo method por valores exatos. |

| Entidade de API       | Campos de localização válidos                                                                                                                                                                                                               | Campos de localização obrigatórios | Valores de campo padrão                                                       | Conteúdo herdável                                                                             |
|-----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------|-------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------|
| Resposta              | <pre>{   "location": {     "type": "RESPONSE"   },   "path":   "<i>resource_path</i> ",   "method":   "<i>http_verb</i> ",   "statusCode":   "<i>status_code</i> " }, ...</pre>                                                             | type                               | Os valores padrão de path, method e statusCode são /, * e *, respectivamente. | Sim, correspondendo path por prefixo e correspondendo method e statusCode por valores exatos. |
| Cabeçalho da resposta | <pre>{   "location": {     "type": "RESPONSE_HEADER",     "path":     "<i>resource_path</i> ",     "method":     "<i>http_verb</i> ",     "statusCode":     "<i>status_code</i> ",     "name":     "<i>header_name</i> "   },   ... }</pre> | type, name                         | Os valores padrão de path, method e statusCode são /, * e *, respectivamente. | Sim, correspondendo path por prefixo e correspondendo method e statusCode por valores exatos. |



| Entidade de API                                  | Campos de localização válidos                                                                                                                                                                            | Campos de localização obrigatórios | Valores de campo padrão                                                       | Conteúdo herdável                                                                             |
|--------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------|-------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------|
| Corpo da resposta                                | <pre>{   "location": {     "type": "RESPONSE_BODY",     "path":       "<i>resource_path</i> ",     "method":       "<i>http_verb</i> ",     "statusCode":       "<i>status_code</i> "   },   ... }</pre> | type                               | Os valores padrão de path, method e statusCode são /, * e *, respectivamente. | Sim, correspondendo path por prefixo e correspondendo method e statusCode por valores exatos. |
| <a href="#">Autorizada</a><br><a href="#">or</a> | <pre>{   "location": {     "type": "AUTHORIZER",     "name":       "<i>authorizer_name</i> "   },   ... }</pre>                                                                                          | type                               | N/D                                                                           | Não                                                                                           |
| <a href="#">Modelo</a>                           | <pre>{   "location": {     "type": "MODEL",     "name":       "<i>model_name</i> "   },   ... }</pre>                                                                                                    | type                               | N/D                                                                           | Não                                                                                           |

## Versões da documentação

Uma versão de documentação é um snapshot da coleção [DocumentationParts](#) de uma API e é marcada com um identificador de versão. Publicar a documentação de uma API envolve criar uma versão da documentação, associá-la a um estágio de API e exportar essa versão específica de estágio da documentação da API para um arquivo do OpenAPI externo. No API Gateway, um snapshot de documentação é representado como um recurso [DocumentationVersion](#).

À medida que você atualiza uma API, novas versões dela são criadas. No API Gateway, todas as versões da documentação são mantidas usando a coleção [DocumentationVersions](#).

## Documentar uma API usando o console do API Gateway

Nesta seção, descrevemos como criar e manter partes da documentação de uma API usando o console do API Gateway.

Um pré-requisito para criar e editar a documentação de uma API é que você já deve ter criado essa API. Nesta seção, usamos a API [PetStore](#) como exemplo. Para criar uma API usando o console do API Gateway, siga as instruções em [Tutorial: Criar uma API REST importando um exemplo](#).

### Tópicos

- [Documentar a entidade API](#)
- [Documentar uma entidade RESOURCE](#)
- [Documentar uma entidade METHOD](#)
- [Documentar uma entidade QUERY\\_PARAMETER](#)
- [Documentar uma entidade PATH\\_PARAMETER](#)
- [Documentar uma entidade REQUEST\\_HEADER](#)
- [Documentar uma entidade REQUEST\\_BODY](#)
- [Documentar uma entidade RESPONSE](#)
- [Documentar uma entidade RESPONSE\\_HEADER](#)
- [Documentar uma entidade RESPONSE\\_BODY](#)
- [Documentar uma entidade MODEL](#)
- [Documentar uma entidade AUTHORIZER](#)

### Documentar a entidade **API**

Para adicionar uma nova parte da documentação da entidade API, faça o seguinte:

1. No painel de navegação principal, selecione Documentação e, depois, Criar parte da documentação.
2. Em Tipo de documentação, selecione API.

Se uma parte de documentação não tiver criada para a API, você acessará o editor de mapa de properties da parte da documentação. Insira o mapa properties a seguir no editor de texto.

```
{
 "info": {
 "description": "Your first API Gateway API.",
 "contact": {
 "name": "John Doe",
 "email": "john.doe@api.com"
 }
 }
}
```

#### Note

Não é necessário codificar o mapa de properties em uma string JSON. O console do API Gateway faz a codificação em string do objeto JSON para você.

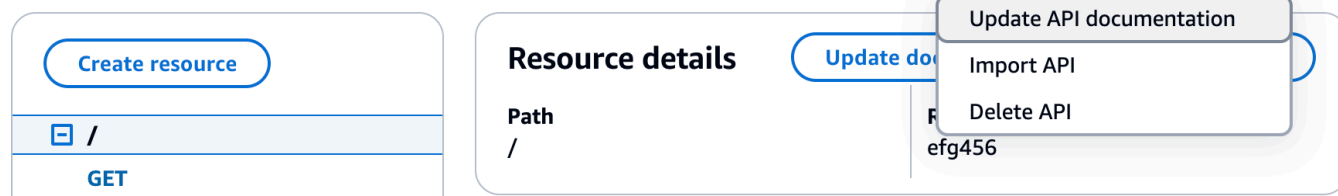
3. Escolha Criar parte da documentação.

Para adicionar uma nova parte da documentação da entidade API no painel Recursos, faça o seguinte:

1. No painel de navegação principal, selecione Recursos.
2. Selecione o menu Ações de API e escolha Atualizar a documentação da API.

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

## Resources



**Create resource**

 /

**GET**

**Resource details**

**Path**  
/

**API actions** ▲

- Update API documentation
- Import API
- Delete API

efg456

Para editar uma parte da documentação existente, faça o seguinte:

1. No painel Documentação, selecione a guia Recursos e métodos.
2. Selecione o nome da API e, no cartão da API, escolha Editar.

## Documentar uma entidade **RESOURCE**

Para adicionar uma nova documentação de uma entidade RESOURCE, faça o seguinte:

1. No painel de navegação principal, selecione Documentação e, depois, Criar parte da documentação.
2. Em Tipo de documentação, selecione Recurso.
3. Em Caminho, insira um caminho.
4. Digite uma descrição no editor de texto; por exemplo:

```
{
 "description": "The PetStore's root resource."
}
```

5. Escolha Criar parte da documentação. É possível criar documentação para um recurso não listado.
6. Se necessário, repita essas etapas para adicionar ou editar outra parte da documentação.

Para adicionar uma nova parte da documentação de uma entidade RESOURCE no painel Recursos, faça o seguinte:

1. No painel de navegação principal, selecione Recursos.
2. Selecione o recurso e escolha Atualizar documentação.

### Resources

The screenshot shows the 'Resources' section of the Amazon API Gateway console. On the left, there is a 'Create resource' button and a list of resources. The first resource is the root path '/' with a 'GET' method. The second resource is '/pets'. On the right, the 'Resource details' for the '/pets' resource are shown. It includes a 'Delete' button, an 'Update documentation' button (which is highlighted with a red rectangle), and an 'Enable CORS' button. The 'Resource ID' is listed as 'bbb111'. At the top right of the Resources section, there are 'API actions' and 'Deploy API' buttons.

Para editar uma parte da documentação existente, faça o seguinte:

1. No painel Documentação, selecione a guia Recursos e métodos.
2. Selecione o recurso que contém a parte da documentação e escolha Editar.

## Documentar uma entidade **METHOD**

Para adicionar uma nova documentação de uma entidade METHOD, faça o seguinte:

1. No painel de navegação principal, selecione Documentação e, depois, Criar parte da documentação.
2. Em Tipo de documentação, selecione Método.
3. Em Caminho, insira um caminho.
4. Em Método, selecione um verbo HTTP.
5. Digite uma descrição no editor de texto; por exemplo:

```
{
 "tags" : ["pets"],
 "summary" : "List all pets"
}
```

6. Escolha Criar parte da documentação. É possível criar documentação para um método não listado.
7. Se necessário, repita essas etapas para adicionar ou editar outra parte da documentação.

Para adicionar uma nova parte da documentação de uma entidade METHOD no painel Recursos, faça o seguinte:

1. No painel de navegação principal, selecione Recursos.
2. Selecione o método e escolha Atualizar documentação.

API Gateway &gt; APIs &gt; Resources - PetStore (abcd1234)

## Resources

**Resources**

API actions ▼ Deploy API

Create resource

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

**/pets - POST - Method execution**

ARN  
arn:aws:execute-api:us-west-2:111222333:abcd1234/\*/POST/pets

Resource ID  
aaa111

Update documentation Delete

Client → Method request → Integration request → HTTP integration

← Method response ← Integration response ←

Para editar uma parte da documentação existente, faça o seguinte:

1. No painel Documentação, selecione a guia Recursos e métodos.
2. É possível selecionar o método ou o recurso que contém o método e usar a barra de pesquisa para localizar e selecionar a parte da documentação.
3. Escolha Editar.

## Documentar uma entidade **QUERY\_PARAMETER**

Para adicionar uma nova documentação de uma entidade **QUERY\_PARAMETER**, faça o seguinte:

1. No painel de navegação principal, selecione Documentação e, depois, Criar parte da documentação.
2. Em Tipo de documentação, selecione Parâmetro de consulta.
3. Em Caminho, insira um caminho.
4. Em Método, selecione um verbo HTTP.
5. Nome, insira um nome.
6. Digite uma descrição no editor de texto.
7. Escolha Criar parte da documentação. É possível criar documentação para um parâmetro de consulta não listado.

8. Se necessário, repita essas etapas para adicionar ou editar outra parte da documentação.

Para editar uma parte da documentação existente, faça o seguinte:

1. No painel Documentação, selecione a guia Recursos e métodos.
2. É possível selecionar o parâmetro de consulta ou o recurso que contém o parâmetro de consulta e usar a barra de pesquisa para localizar e selecionar a parte da documentação.
3. Escolha Editar.

### Documentar uma entidade **PATH\_PARAMETER**

Para adicionar uma nova documentação de uma entidade PATH\_PARAMETER, faça o seguinte:

1. No painel de navegação principal, selecione Documentação e, depois, Criar parte da documentação.
2. Em Tipo de documentação, selecione Parâmetro de caminho.
3. Em Caminho, insira um caminho.
4. Em Método, selecione um verbo HTTP.
5. Nome, insira um nome.
6. Digite uma descrição no editor de texto.
7. Escolha Criar parte da documentação. É possível criar documentação para um parâmetro de caminho não listado.
8. Se necessário, repita essas etapas para adicionar ou editar outra parte da documentação.

Para editar uma parte da documentação existente, faça o seguinte:

1. No painel Documentação, selecione a guia Recursos e métodos.
2. É possível selecionar o parâmetro de caminho ou o recurso que contém o parâmetro de caminho e usar a barra de pesquisa para localizar e selecionar a parte da documentação.
3. Escolha Editar.

### Documentar uma entidade **REQUEST\_HEADER**

Para adicionar uma nova documentação de uma entidade REQUEST\_HEADER, faça o seguinte:

1. No painel de navegação principal, selecione Documentação e, depois, Criar parte da documentação.
2. Em Tipo de documentação, selecione Cabeçalho da solicitação.
3. Em Caminho, insira um caminho para o cabeçalho da solicitação.
4. Em Método, selecione um verbo HTTP.
5. Nome, insira um nome.
6. Digite uma descrição no editor de texto.
7. Escolha Criar parte da documentação. É possível criar documentação para um cabeçalho da solicitação não listado.
8. Se necessário, repita essas etapas para adicionar ou editar outra parte da documentação.

Para editar uma parte da documentação existente, faça o seguinte:

1. No painel Documentação, selecione a guia Recursos e métodos.
2. É possível selecionar o cabeçalho da solicitação ou o recurso que contém o cabeçalho e usar a barra de pesquisa para localizar e selecionar a parte da documentação.
3. Escolha Editar.

## Documentar uma entidade **REQUEST\_BODY**

Para adicionar uma nova documentação de uma entidade REQUEST\_BODY, faça o seguinte:

1. No painel de navegação principal, selecione Documentação e, depois, Criar parte da documentação.
2. Em Tipo de documentação, selecione Corpo da solicitação.
3. Em Caminho, insira um caminho para o corpo da solicitação.
4. Em Método, selecione um verbo HTTP.
5. Digite uma descrição no editor de texto.
6. Escolha Criar parte da documentação. É possível criar documentação para um corpo da solicitação não listado.
7. Se necessário, repita essas etapas para adicionar ou editar outra parte da documentação.

Para editar uma parte da documentação existente, faça o seguinte:



1. No painel Documentação, selecione a guia Recursos e métodos.
2. É possível selecionar o corpo da solicitação ou o recurso que contém o corpo e usar a barra de pesquisa para localizar e selecionar a parte da documentação.
3. Escolha Editar.

## Documentar uma entidade **RESPONSE**

Para adicionar uma nova documentação de uma entidade RESPONSE, faça o seguinte:

1. No painel de navegação principal, selecione Documentação e, depois, Criar parte da documentação.
2. Em Tipo de documentação, selecione Resposta (código de status).
3. Em Caminho, insira um caminho para a resposta.
4. Em Método, selecione um verbo HTTP.
5. Em Código de status, insira um código de status HTTP.
6. Digite uma descrição no editor de texto.
7. Escolha Criar parte da documentação. É possível criar documentação para um status de resposta não listado.
8. Se necessário, repita essas etapas para adicionar ou editar outra parte da documentação.

Para editar uma parte da documentação existente, faça o seguinte:

1. No painel Documentação, selecione a guia Recursos e métodos.
2. É possível selecionar o código de status da resposta ou o recurso que contém o código e usar a barra de pesquisa para localizar e selecionar a parte da documentação.
3. Escolha Editar.

## Documentar uma entidade **RESPONSE\_HEADER**

Para adicionar uma nova documentação de uma entidade RESPONSE\_HEADER, faça o seguinte:

1. No painel de navegação principal, selecione Documentação e, depois, Criar parte da documentação.
2. Em Tipo de documentação, selecione Cabeçalho da resposta.

3. Em Caminho, insira um caminho para o cabeçalho da resposta.
4. Em Método, selecione um verbo HTTP.
5. Em Código de status, insira um código de status HTTP.
6. Digite uma descrição no editor de texto.
7. Escolha Criar parte da documentação. É possível criar documentação para um cabeçalho da resposta não listado.
8. Se necessário, repita essas etapas para adicionar ou editar outra parte da documentação.

Para editar uma parte da documentação existente, faça o seguinte:

1. No painel Documentação, selecione a guia Recursos e métodos.
2. É possível selecionar o cabeçalho da resposta ou o recurso que contém o cabeçalho e usar a barra de pesquisa para localizar e selecionar a parte da documentação.
3. Escolha Editar.

## Documentar uma entidade **RESPONSE\_BODY**

Para adicionar uma nova documentação de uma entidade RESPONSE\_BODY, faça o seguinte:

1. No painel de navegação principal, selecione Documentação e, depois, Criar parte da documentação.
2. Em Tipo de documentação, selecione Corpo da resposta.
3. Em Caminho, insira um caminho para o corpo da resposta.
4. Em Método, selecione um verbo HTTP.
5. Em Código de status, insira um código de status HTTP.
6. Digite uma descrição no editor de texto.
7. Escolha Criar parte da documentação. É possível criar documentação para um corpo da resposta não listado.
8. Se necessário, repita essas etapas para adicionar ou editar outra parte da documentação.

Para editar uma parte da documentação existente, faça o seguinte:

1. No painel Documentação, selecione a guia Recursos e métodos.

2. É possível selecionar o corpo da resposta ou o recurso que contém o corpo e usar a barra de pesquisa para localizar e selecionar a parte da documentação.
3. Escolha Editar.

## Documentar uma entidade **MODEL**

A documentação de uma entidade MODEL envolve a criação e o gerenciamento de instâncias de `DocumentPart` para o modelo e cada uma das `properties` do modelo. Por exemplo, o modelo `Error` que acompanha cada API por padrão tem a seguinte definição de esquema,

```
{
 "$schema" : "http://json-schema.org/draft-04/schema#",
 "title" : "Error Schema",
 "type" : "object",
 "properties" : {
 "message" : { "type" : "string" }
 }
}
```

e requer duas instâncias de `DocumentationPart`, uma para o `Model` e a outra para sua propriedade `message`:

```
{
 "location": {
 "type": "MODEL",
 "name": "Error"
 },
 "properties": {
 "title": "Error Schema",
 "description": "A description of the Error model"
 }
}
```

e

```
{
 "location": {
 "type": "MODEL",
 "name": "Error.message"
 },
 "properties": {
```

```
"description": "An error message."
}
```

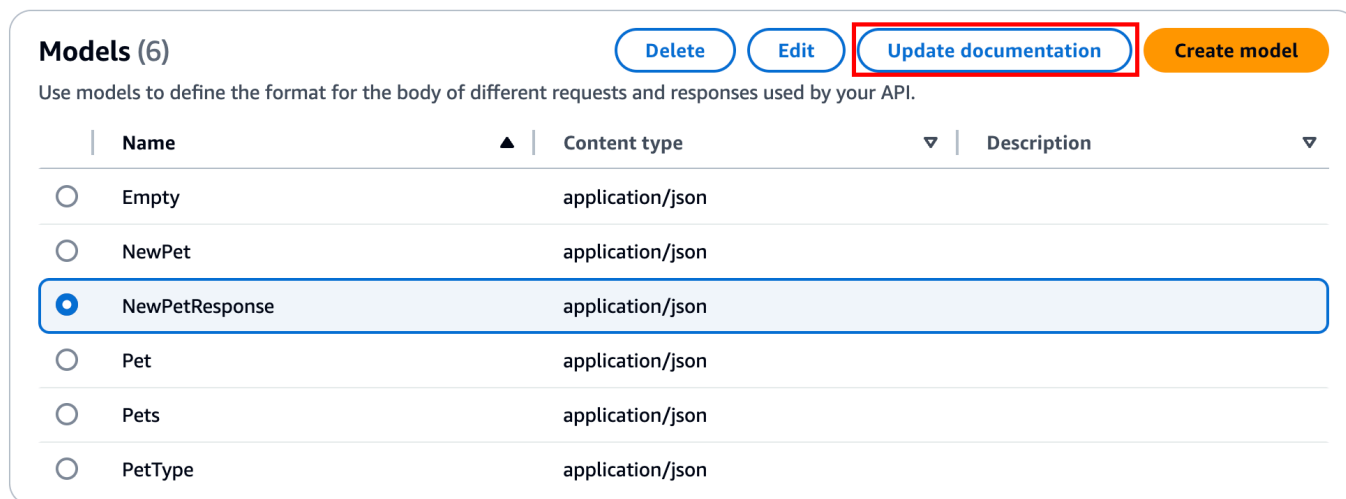
Quando a API for exportada, as propriedades de `DocumentationPart` substituirão os valores no esquema original.

Para adicionar uma nova documentação de uma entidade `MODEL`, faça o seguinte:

1. No painel de navegação principal, selecione Documentação e, depois, Criar parte da documentação.
2. Em Tipo de documentação, selecione Modelo.
3. Em Nome, insira um nome para o modelo.
4. Digite uma descrição no editor de texto.
5. Escolha Criar parte da documentação. É possível criar documentação para modelos não listados.
6. Se necessário, repita essas etapas para editar ou adicionar uma parte de documentação a outros modelos.

Para adicionar uma nova parte da documentação de uma entidade `MODEL` no painel Modelos, faça o seguinte:

1. No painel de navegação principal, selecione Modelos.
2. Selecione o modelo e escolha Atualizar documentação.



**Models (6)** [Delete](#) [Edit](#) [Update documentation](#) [Create model](#)

Use models to define the format for the body of different requests and responses used by your API.

|                                  | Name           | Content type     | Description |
|----------------------------------|----------------|------------------|-------------|
| <input type="radio"/>            | Empty          | application/json |             |
| <input type="radio"/>            | NewPet         | application/json |             |
| <input checked="" type="radio"/> | NewPetResponse | application/json |             |
| <input type="radio"/>            | Pet            | application/json |             |
| <input type="radio"/>            | Pets           | application/json |             |
| <input type="radio"/>            | PetType        | application/json |             |

Para editar uma parte da documentação existente, faça o seguinte:

1. No painel Documentação, selecione a guia Modelos.
2. Use a barra de pesquisa ou selecione o modelo e, depois, escolha Editar.

## Documentar uma entidade **AUTHORIZER**

Para adicionar uma nova documentação de uma entidade AUTHORIZER, faça o seguinte:

1. No painel de navegação principal, selecione Documentação e, depois, Criar parte da documentação.
2. Em Tipo de documentação, selecione Autorizador.
3. Em Nome, insira o nome do autorizador.
4. Digite uma descrição no editor de texto. Especifique um valor para o campo `location` válido para o autorizador.
5. Escolha Criar parte da documentação. É possível criar documentação para autorizadores não listados.
6. Se necessário, repita essas etapas para editar ou adicionar uma parte de documentação a outros autorizadores.

Para editar uma parte da documentação existente, faça o seguinte:

1. No painel Documentação, selecione a guia Autorizadores.
2. Use a barra de pesquisa ou selecione o autorizador e, depois, escolha Editar.

## Publicar a documentação da API usando o console do API Gateway

O procedimento a seguir descreve como publicar uma versão de documentação.

Como publicar uma versão de documentação usando o console do API Gateway

1. No painel de navegação principal, escolha Documentação.
2. Selecione Publicar documentação.
3. Configure a publicação:
  - a. Em Estágio, selecione um estágio.

- b. Em Versão, insira um identificador de versão; por exemplo, 1.0.0.
  - c. (Opcional) Em Description (Descrição), insira uma descrição.
4. Escolha Publish.

Agora, você pode prosseguir e fazer download da documentação publicada, exportando a documentação para um arquivo do OpenAPI externo. Para saber mais, consulte [the section called “Exportar uma API REST”](#).

## Documentar uma API usando a API REST do API Gateway

Nesta seção, descrevemos como criar e manter partes da documentação de uma API usando a API REST do API Gateway.

Crie uma API antes de criar e editar sua documentação. Nesta seção, usamos a API [PetStore](#) como exemplo. Para criar uma API usando o console do API Gateway, siga as instruções em [Tutorial: Criar uma API REST importando um exemplo](#).

### Tópicos

- [Documentar a entidade API](#)
- [Documentar uma entidade RESOURCE](#)
- [Documentar uma entidade METHOD](#)
- [Documentar uma entidade QUERY\\_PARAMETER](#)
- [Documentar uma entidade PATH\\_PARAMETER](#)
- [Documentar uma entidade REQUEST\\_BODY](#)
- [Documentar uma entidade REQUEST\\_HEADER](#)
- [Documentar uma entidade RESPONSE](#)
- [Documentar uma entidade RESPONSE\\_HEADER](#)
- [Documentar uma entidade AUTHORIZER](#)
- [Documentar uma entidade MODEL](#)
- [Atualizar partes da documentação](#)
- [Listar partes da documentação](#)

## Documentar a entidade **API**

Para adicionar documentação para uma [API](#), adicione um recurso [DocumentationPart](#) à entidade 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\t}\n}"
}
```

Se bem-sucedida, a operação retornará uma resposta 201 Created, contendo a instância DocumentationPart recém-criada na carga útil. Por exemplo:

```
{
 ...
 "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\t}\n}"
}
```

Se a parte de documentação já tiver sido adicionada, será retornada uma resposta 409 Conflict contendo a mensagem de erro Documentation part already exists for the specified location: type 'API'.". Nesse caso, você deve chamar a operação [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}"
 }]
}

```

A resposta bem-sucedida retorna um código de status 200 OK com a carga útil que contém a instância `DocumentationPart` atualizada.

## Documentar uma entidade **RESOURCE**

Para adicionar a documentação do recurso raiz de uma API, adicione um recurso [DocumentationPart](#) direcionado ao recurso [Resource](#) correspondente:

```

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\t}"
}

```

Se bem-sucedida, a operação retornará uma resposta 201 Created, contendo a instância `DocumentationPart` recém-criada na carga útil. Por exemplo:



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

Quando o caminho do recurso não é especificado, supõe-se que esse recurso seja raiz. Você pode adicionar "path": "/" a properties para tornar a especificação explícita.

Para criar a documentação de um recurso filho de uma API, adicione um recurso [DocumentationPart](#) direcionado ao recurso [Resource](#) correspondente:

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret
```

```
{
 "location" : {
 "type" : "RESOURCE",
 "path" : "/pets"
 },
 "properties": "{\n\t\"description\" : \"A child resource under the root of
PetStore.\n\n}"
}
```

Se bem-sucedida, a operação retornará uma resposta 201 Created, contendo a instância `DocumentationPart` recém-criada na carga útil. Por exemplo:

```
{
 "_links": {
 "curies": {
 "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
 "name": "documentationpart",
 "templated": true
 },
 "self": {
 "href": "/restapis/4wk1k4onj3/documentation/parts/qcht86"
 },
 "documentationpart:delete": {
 "href": "/restapis/4wk1k4onj3/documentation/parts/qcht86"
 },
 "documentationpart:update": {
 "href": "/restapis/4wk1k4onj3/documentation/parts/qcht86"
 }
 },
 "id": "qcht86",
 "location": {
 "path": "/pets",
 "method": null,
 "name": null,
 "statusCode": null,
 "type": "RESOURCE"
 },
 "properties": "{\n\t\"description\" : \"A child resource under the root of PetStore.
\n\n}"
}
```

Para adicionar a documentação de um recurso filho especificado por um parâmetro de caminho, adicione um recurso [DocumentationPart](#) direcionado ao recurso [Resource](#):

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
 "location" : {
 "type" : "RESOURCE",
 "path" : "/pets/{petId}"
 },
 "properties": "{\n\t\"description\" : \"A child resource specified by the petId
path parameter.\\n\\n}"
}
```

Se bem-sucedida, a operação retornará uma resposta 201 Created, contendo a instância `DocumentationPart` recém-criada na carga útil. Por exemplo:

```
{
 "_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\n}"
}
```

### Note

A instância de [DocumentationPart](#) de uma entidade RESOURCE não pode ser herdada por nenhum de seus recursos filho.

## Documentar uma entidade **METHOD**

Para adicionar a documentação para um método de uma API, adicione um recurso [DocumentationPart](#) direcionado ao recurso [Method](#) correspondente:

```

POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTtttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
 "location" : {
 "type" : "METHOD",
 "path" : "/pets",
 "method" : "GET"
 },
 "properties": "{\n\t\"summary\" : \"List all pets.\n\n}"
}
```

Se bem-sucedida, a operação retornará uma resposta 201 Created, contendo a instância DocumentationPart recém-criada na carga útil. Por exemplo:

```
{
```

```

"_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}"
}

```

Se bem-sucedida, a operação retornará uma resposta 201 Created, contendo a instância DocumentationPart recém-criada na carga útil. Por exemplo:

```

{
 "_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\n}"
}

```

Se o campo `location.method` não for especificado na solicitação anterior, supõe-se que ele seja o método ANY que é representado por um caractere curinga `*`.

Para atualizar o conteúdo de documentação de uma entidade METHOD, chame a operação [documentationpart:update](#), fornecendo um novo mapa de properties:

```

PATCH /restapis/4wk1k4onj3/documentation/parts/part_id HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTtttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
 "patchOperations" : [{
 "op" : "replace",
 "path" : "/properties",
 "value" : "{\n\t\"tags\" : [\"pets\"], \n\t\"summary\" : \"List all pets.\n\n}"
 }]
}

```

A resposta bem-sucedida retorna um código de status 200 OK com a carga útil que contém a instância `DocumentationPart` atualizada. Por exemplo:

```
{
 "_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 uma entidade **QUERY\_PARAMETER**

Para adicionar a documentação de um parâmetro de consulta de solicitação, adicione um recurso [DocumentationPart](#) direcionado ao tipo **QUERY\_PARAMETER**, com os campos válidos de path e name.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTtttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret
```

```
{
 "location" : {
 "type" : "QUERY_PARAMETER",
 "path" : "/pets",
 "method" : "GET",
 "name" : "page"
 },
 "properties": "{\n\t\"description\" : \"Page number of results to return.\"\n}"
}
```

Se bem-sucedida, a operação retornará uma resposta 201 Created, contendo a instância `DocumentationPart` recém-criada na carga útil. Por exemplo:

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



O mapa de properties da parte de documentação de uma entidade QUERY\_PARAMETER pode ser herdado por uma de suas entidades QUERY\_PARAMETER filho. Por exemplo, se você adicionar um recurso `treats` depois de `/pets/{petId}`, habilitar o método GET em `/pets/{petId}/treats` e expor o parâmetro de consulta `page`, esse parâmetro de consulta filho herdará o mapa `DocumentationPart` de properties do parâmetro de consulta com nome semelhante do método GET `/pets`, a menos que você adicione explicitamente um recurso `DocumentationPart` ao parâmetro de consulta `page` do método GET `/pets/{petId}/treats`.

## Documentar uma entidade PATH\_PARAMETER

Para adicionar a documentação de um parâmetro de caminho, adicione um recurso [DocumentationPart](#) para a entidade PATH\_PARAMETER.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTtttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
 "location" : {
 "type" : "PATH_PARAMETER",
 "path" : "/pets/{petId}",
 "method" : "*",
 "name" : "petId"
 },
 "properties": "{\n\t\"description\" : \"The id of the pet to retrieve.\n\n}"
}
```

Se bem-sucedida, a operação retornará uma resposta 201 Created, contendo a instância `DocumentationPart` recém-criada na carga útil. Por exemplo:

```
{
 "_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 uma entidade **REQUEST\_BODY**

Para adicionar a documentação de um corpo de solicitação, adicione um recurso [DocumentationPart](#) para o corpo da solicitação.

```

POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTtttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
 "location" : {
 "type" : "REQUEST_BODY",
 "path" : "/pets",
 "method" : "POST"
 },
 "properties": "{\n\t\"description\" : \"A Pet object to be added to PetStore.\"\n}"
}
```

Se bem-sucedida, a operação retornará uma resposta 201 Created, contendo a instância `DocumentationPart` recém-criada na carga útil. Por exemplo:

```
{
 "_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 uma entidade **REQUEST\_HEADER**

Para adicionar a documentação de um cabeçalho de solicitação, adicione um recurso [DocumentationPart](#) para o cabeçalho da solicitação.

```
POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
```

```
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret
```

```
{
 "location" : {
 "type" : "REQUEST_HEADER",
 "path" : "/pets",
 "method" : "GET",
 "name" : "x-my-token"
 },
 "properties": "{\n\t\"description\" : \"A custom token used to authorization the
method invocation.\n\n}"
}
```

Se bem-sucedida, a operação retornará uma resposta 201 Created, contendo a instância `DocumentationPart` recém-criada na carga útil. Por exemplo:

```
{
 "_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\n\tmethod invocation.\\\"\\n}\"
 }

```

## Documentar uma entidade **RESPONSE**

Para adicionar a documentação para uma resposta de um código de status, adicione um recurso [DocumentationPart](#) direcionado para o recurso [MethodResponse](#) correspondente.

```

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\"
}

```

Se bem-sucedida, a operação retornará uma resposta 201 Created, contendo a instância `DocumentationPart` recém-criada na carga útil. Por exemplo:

```

{
 "_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": "lattice",
 "location": {
 "path": "/",
 "method": "*",
 "name": null,
 "statusCode": "200",
 "type": "RESPONSE"
 },
 "properties": "{\n \"description\" : \"Successful operation.\\n\\n\"
 }

```

## Documentar uma entidade **RESPONSE\_HEADER**

Para adicionar a documentação de um cabeçalho de resposta, adicione um recurso [DocumentationPart](#) para o cabeçalho da solicitação.

```

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\\n\"

```

Se bem-sucedida, a operação retornará uma resposta 201 Created, contendo a instância `DocumentationPart` recém-criada na carga útil. Por exemplo:

```

{
 "_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}"
}

```

A documentação desse cabeçalho de resposta Content-Type é a documentação padrão para os cabeçalhos Content-Type de todas as respostas da API.

## Documentar uma entidade **AUTHORIZER**

Para adicionar a documentação de um autorizador de API, adicione um recurso [DocumentationPart](#) direcionado ao autorizador especificado.

```

POST /restapis/restapi_id/documentation/parts HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTtttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
 "location" : {
 "type" : "AUTHORIZER",
 "name" : "myAuthorizer"
 }
}

```

```

 },
 "properties": "{\n\t\"description\" : \"Authorizes invocations of configured
methods.\n\n}"
}

```

Se bem-sucedida, a operação retornará uma resposta 201 Created, contendo a instância DocumentationPart recém-criada na carga útil. Por exemplo:

```

{
 "_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\n}"
}

```



 Note

A instância de [DocumentationPart](#) de uma entidade AUTHORIZER não pode ser herdada por nenhum de seus recursos filho.

## Documentar uma entidade **MODEL**

A documentação de uma entidade MODEL envolve a criação e o gerenciamento de instâncias de DocumentPart para o modelo e cada uma das properties do modelo. Por exemplo, o modelo Error que acompanha cada API por padrão tem a seguinte definição de esquema,

```
{
 "$schema" : "http://json-schema.org/draft-04/schema#",
 "title" : "Error Schema",
 "type" : "object",
 "properties" : {
 "message" : { "type" : "string" }
 }
}
```

e requer duas instâncias de DocumentationPart, uma para o Model e a outra para sua propriedade message:

```
{
 "location": {
 "type": "MODEL",
 "name": "Error"
 },
 "properties": {
 "title": "Error Schema",
 "description": "A description of the Error model"
 }
}
```

e

```
{
 "location": {
 "type": "MODEL",
 "name": "Error.message"
 }
```

```

},
"properties": {
 "description": "An error message."
}
}

```

Quando a API for exportada, as propriedades de `DocumentationPart` substituirão os valores no esquema original.

Para adicionar a documentação de um modelo de API, adicione um recurso [DocumentationPart](#) direcionado ao 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}"
}

```

Se bem-sucedida, a operação retornará uma resposta 201 Created, contendo a instância `DocumentationPart` recém-criada na carga útil. Por exemplo:

```

{
 "_links": {
 "curies": {
 "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-
documentationpart-{rel}.html",
 "name": "documentationpart",
 "templated": true
 },
 "self": {
 "href": "/restapis/4wk1k4onj3/documentation/parts/1kn4uq"
 }
 },

```

```

 "documentationpart:delete": {
 "href": "/restapis/4wk1k4onj3/documentation/parts/1kn4uq"
 },
 "documentationpart:update": {
 "href": "/restapis/4wk1k4onj3/documentation/parts/1kn4uq"
 }
 },
 "id": "1kn4uq",
 "location": {
 "path": null,
 "method": null,
 "name": "Pet",
 "statusCode": null,
 "type": "MODEL"
 },
 "properties": "{\n\t\"description\" : \"Data structure of a Pet object.\"\n}"
}

```

Repita a mesma etapa para criar uma instância de `DocumentationPart` para qualquer uma das propriedades do modelo.

#### Note

A instância de [DocumentationPart](#) de uma entidade `MODEL` não pode ser herdada por nenhum de seus recursos filho.

## Atualizar partes da documentação

Para atualizar as partes de documentação de qualquer tipo de entidade de API, envie uma solicitação `PATCH` em uma instância de [DocumentationPart](#) de um identificador de parte especificado para substituir o mapa de `properties` existente por um novo.

```

PATCH /restapis/4wk1k4onj3/documentation/parts/part_id HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTtttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{

```

```
"patchOperations" : [{
 "op" : "replace",
 "path" : "RESOURCE_PATH",
 "value" : "NEW_properties_VALUE_AS_JSON_STRING"
}]
}
```

A resposta bem-sucedida retorna um código de status 200 OK com a carga útil que contém a instância `DocumentationPart` atualizada.

Você pode atualizar várias partes de documentação em uma única solicitação PATCH.

### Listar partes da documentação

Para listar as partes de documentação de qualquer tipo de entidade de API, envie uma solicitação GET em uma coleção [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
```

A resposta bem-sucedida retorna um código de status 200 OK com a carga útil que contém as instâncias `DocumentationPart` disponíveis.

### Publicar a documentação da API usando a API REST do API Gateway

Para publicar a documentação de uma API, crie, atualize ou obtenha um snapshot de documentação e depois associe esse snapshot a um estágio de API. Ao criar um snapshot de documentação, você também pode associá-lo a um estágio de API ao mesmo tempo.

### Tópicos

- [Criar um snapshot de documentação e associá-lo a um estágio de API](#)
- [Criar um snapshot de documentação](#)
- [Atualizar um snapshot de documentação](#)
- [Obter um snapshot de documentação](#)
- [Associar um snapshot de documentação a um estágio de API](#)

- [Fazer download de um snapshot de documentação associado a um estágio](#)

## Criar um snapshot de documentação e associá-lo a um estágio de API

Para criar um snapshot das partes de documentação de uma API e associá-lo a um estágio de API ao mesmo tempo, envie a seguinte solicitação POST:

```
POST /restapis/restapi_id/documentation/versions HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTtttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
 "documentationVersion" : "1.0.0",
 "stageName": "prod",
 "description" : "My API Documentation v1.0.0"
}
```

Se bem-sucedida, a operação retorna uma resposta 200 OK, contendo a instância recentemente DocumentationVersion criada como a carga útil.

Como alternativa, você pode criar um snapshot de documentação sem associá-lo a um estágio da API primeiro e, depois, chamar [restapi:update](#) para associar esse snapshot a um estágio da API especificado. Você também pode atualizar ou consultar um snapshot de documentação existente e depois atualizar sua associação de estágio. Mostramos as etapas nas próximas quatro seções.

## Criar um snapshot de documentação

Para criar um snapshot de partes de documentação de uma API, crie um recurso [DocumentationVersion](#) e adicione-o à coleção [DocumentationVersions](#) da API:

```
POST /restapis/restapi_id/documentation/versions HTTP/1.1
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTtttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret
```

```
{
 "documentationVersion" : "1.0.0",
 "description" : "My API Documentation v1.0.0"
}
```

Se bem-sucedida, a operação retorna uma resposta 200 OK, contendo a instância recentemente `DocumentationVersion` criada como a carga útil.

### Atualizar um snapshot de documentação

Você só pode atualizar um snapshot de documentação modificando a propriedade `description` do recurso [DocumentationVersion](#) correspondente. O exemplo a seguir mostra como atualizar a descrição do snapshot de documentação, conforme identificado por seu identificador de versão, *version*, por exemplo, 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: YYYYMMDDTtttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
 "patchOperations": [{
 "op": "replace",
 "path": "/description",
 "value": "My API for testing purposes."
 }]
}
```

Se bem-sucedida, a operação retornará uma resposta 200 OK, contendo a instância `DocumentationVersion` atualizada como a carga útil.

### Obter um snapshot de documentação

Para obter um snapshot de documentação, envie uma solicitação GET com base no recurso [DocumentationVersion](#) especificado. O exemplo a seguir mostra como obter um snapshot de documentação de um determinado identificador de versão, 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
```

## Associar um snapshot de documentação a um estágio de API

Para publicar a documentação da API, associe um snapshot de documentação a um estágio de API. Você já deve ter criado um estágio de API antes de associar a versão da documentação ao estágio.

Para associar um snapshot de documentação a um estágio de API usando a [API REST do API Gateway](#), chame a operação [stage:update](#) para definir a versão de documentação desejada na propriedade `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"
 }]
}
```

## Fazer download de um snapshot de documentação associado a um estágio

Depois que uma versão das partes da documentação é associada a um estágio, você pode exportar as partes da documentação junto com as definições de entidades de API para um arquivo externo, usando o console do API Gateway, a API REST do API Gateway, um dos SDKs ou a AWS CLI para o API Gateway. O processo é o mesmo para a exportação da API. O formato do arquivo exportado pode ser JSON ou YAML.

Usando a API REST do API Gateway, você também pode definir explicitamente o parâmetro de consulta `extension=documentation,integrations,authorizers` para incluir as partes de documentação da API, as integrações da API e os autorizadores em uma exportação de API. Por

padrão, partes de documentação são incluídas, mas integrações e autorizadores são excluídos, quando você exporta uma API. A saída padrão de uma exportação de API é adequada para a distribuição da documentação.

Para exportar a documentação da API em um arquivo do OpenAPI JSON externo usando a API REST do API Gateway, envie a seguinte solicitação GET:

```
GET /restapis/restapi_id/stages/stage_name/exports/swagger?extensions=documentation
HTTP/1.1
Accept: application/json
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTtttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret
```

Aqui, o objeto `x-amazon-apigateway-documentation` contém as partes de documentação, e as definições da entidade de API contém as propriedades de documentação com suporte pelo OpenAPI. A saída não inclui detalhes de integração nem dos autorizadores do Lambda (anteriormente conhecidos como autorizadores personalizados). Para incluir ambos os detalhes, defina `extensions=integrations,authorizers,documentation`. Para incluir detalhes de integrações, mas não de autorizadores, defina `extensions=integrations,documentation`.

Você deve definir o cabeçalho `Accept: application/json` na solicitação para processar a saída do resultado em um arquivo JSON. Para produzir a saída YAML, altere o cabeçalho da solicitação para `Accept: application/yaml`.

Como exemplo, vamos observar uma API que expõe um método simples GET no recurso raiz (/). Essa API tem quatro entidades de API definidas em um arquivo de definição do OpenAPI, uma para cada um dos tipos API, MODEL, METHOD e RESPONSE. Uma parte de documentação foi adicionada a cada uma das entidades API, METHOD e RESPONSE. Chamando o comando de exportação de documentação anterior, obtemos a seguinte saída, com as partes de documentação listadas dentro do objeto `x-amazon-apigateway-documentation` como uma extensão de um arquivo padrão do OpenAPI.

## 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 um atributo compatível com o OpenAPI definido no mapa de `properties` de uma parte de documentação, o API Gateway insere esse atributo na definição de entidade de API associada. Um atributo de `x-something` é uma extensão do OpenAPI padrão. Essa extensão é propagada na definição da entidade de API. Por exemplo, consulte o atributo `x-example` do método GET. Um atributo como `foo` não faz parte da especificação OpenAPI e não é injetado em suas definições de entidade de API associadas.

Se uma ferramenta de renderização de documentação (por exemplo, a [interface do OpenAPI](#)) analisar as definições de entidades de API para extrair atributos de documentação, nenhum dos atributos de `properties` não compatíveis com o OpenAPI de uma instância de `DocumentationPart` estará disponível para a ferramenta. No entanto, se uma ferramenta de renderização de documentação analisar o objeto `x-amazon-apigateway-documentation` para obter conteúdo, ou se a ferramenta chamar [restapi:documentation-parts](#) e [documentationpart:by-id](#) para recuperar partes da documentação do API Gateway, todos os atributos da documentação estarão disponíveis para a ferramenta exibir.

Para exportar a documentação com as definições de entidades de API contendo detalhes de integração para um arquivo JSON do OpenAPI, envie a seguinte solicitação 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 a documentação com as definições de entidades de API contendo detalhes de integrações e autorizadores para um arquivo YAML do OpenAPI, envie a seguinte solicitação 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 usar o console do API Gateway para exportar e fazer download da documentação publicada de uma API, siga as instruções em [Exportar API REST usando o console do API Gateway](#).

## Importar a documentação da API

Como na importação de definições de entidades de API, você pode importar partes de documentação de um arquivo externo do OpenAPI para uma API no API Gateway. Especifique as partes de documentação a serem importadas dentro da extensão [Objeto x-amazon-apigateway-documentation](#) em um arquivo de definição do OpenAPI válido. Importar a documentação não altera as definições de entidades de API existentes.

Você tem a opção de mesclar as partes de documentação recém-especificadas em partes de documentação existentes no API Gateway ou de substituir as partes de documentação existentes. No modo MERGE, uma nova parte de documentação definida no arquivo do OpenAPI é adicionada à coleção DocumentationParts da API. Se uma DocumentationPart importada já existir, um atributo importado substituirá o existente caso os dois sejam diferentes. Outros atributos de documentação existentes permanecem inalterados. No modo OVERWRITE, a coleção DocumentationParts inteira é substituída de acordo com o arquivo de definição do OpenAPI importado.

## Importar partes de documentação com a API REST do API Gateway

Para importar a documentação da API usando a API REST do API Gateway, chame a operação [documentationpart:import](#). O exemplo a seguir mostra como substituir partes de documentação existentes de uma API por um único método GET / , retornando uma resposta 200 OK em caso de êxito.

### OpenAPI 3.0

```
PUT /restapis/<restapi_id>/documentation/parts&mode=overwrite&failonwarnings=true
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTtttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
 "openapi": "3.0.0",
 "info": {
 "description": "description",
 "version": "1",
 "title": "doc"
 },
 "paths": {
 "/": {
 "get": {
 "description": "Method description.",
 "responses": {
 "200": {
 "description": "200 response",
 "content": {
 "application/json": {
 "schema": {
 "$ref": "#/components/schemas/Empty"
 }
 }
 }
 }
 }
 }
 }
 }
},
```

```
"x-amazon-apigateway-documentation": {
 "version": "1.0.3",
 "documentationParts": [
 {
 "location": {
 "type": "API"
 },
 "properties": {
 "description": "API description",
 "info": {
 "description": "API info description 4",
 "version": "API info version 3"
 }
 }
 },
 {
 "location": {
 "type": "METHOD",
 "method": "GET"
 },
 "properties": {
 "description": "Method description."
 }
 },
 {
 "location": {
 "type": "MODEL",
 "name": "Empty"
 },
 "properties": {
 "title": "Empty Schema"
 }
 },
 {
 "location": {
 "type": "RESPONSE",
 "method": "GET",
 "statusCode": "200"
 },
 "properties": {
 "description": "200 response"
 }
 }
]
}
```



```

 },
 "servers": [
 {
 "url": "/"
 }
],
 "components": {
 "schemas": {
 "Empty": {
 "type": "object",
 "title": "Empty Schema"
 }
 }
 }
 }
}

```

## OpenAPI 2.0

```

PUT /restapis/<restapi_id>/documentation/parts&mode=overwrite&failonwarnings=true
Host: apigateway.region.amazonaws.com
Content-Type: application/json
X-Amz-Date: YYYYMMDDTttttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

```

```

{
 "swagger": "2.0",
 "info": {
 "description": "description",
 "version": "1",
 "title": "doc"
 },
 "host": "",
 "basePath": "/",
 "schemes": [
 "https"
],
 "paths": {
 "/": {
 "get": {
 "description": "Method description.",
 "produces": [

```

```
 "application/json"
],
 "responses": {
 "200": {
 "description": "200 response",
 "schema": {
 "$ref": "#/definitions/Empty"
 }
 }
 }
 }
 },
 "definitions": {
 "Empty": {
 "type": "object",
 "title": "Empty Schema"
 }
 },
 "x-amazon-apigateway-documentation": {
 "version": "1.0.3",
 "documentationParts": [
 {
 "location": {
 "type": "API"
 },
 "properties": {
 "description": "API description",
 "info": {
 "description": "API info description 4",
 "version": "API info version 3"
 }
 }
 },
 {
 "location": {
 "type": "METHOD",
 "method": "GET"
 },
 "properties": {
 "description": "Method description."
 }
 }
]
 },
 {
```

```

 "location": {
 "type": "MODEL",
 "name": "Empty"
 },
 "properties": {
 "title": "Empty Schema"
 }
 },
 {
 "location": {
 "type": "RESPONSE",
 "method": "GET",
 "statusCode": "200"
 },
 "properties": {
 "description": "200 response"
 }
 }
]
}
}

```

Quando bem-sucedida, essa solicitação retorna uma resposta 200 OK contendo o `DocumentationPartId` importado na carga.

```

{
 "ids": [
 "kg3mth",
 "796rtf",
 "zhek4p",
 "5ukm9s"
]
}

```

Além disso, você também pode chamar [restapi:import](#) ou [restapi:put](#), fornecendo as partes de documentação no objeto `x-amazon-apigateway-documentation` como parte do arquivo de entrada do OpenAPI da definição da API. Para excluir as partes de documentação da importação da API, defina `ignore=documentation` nos parâmetros da consulta de solicitação.

Importar partes de documentação com o console do API Gateway

As instruções a seguir descrevem como importar partes de documentação.

Para usar o console de modo a importar partes de documentação de uma API a partir de um arquivo externo

1. No painel de navegação principal, escolha Documentação.
2. Escolha Importar.
3. Se você já tiver uma documentação, selecione Substituir ou Mesclar a nova documentação.
4. Selecione Escolher arquivo para carregar um arquivo de uma unidade ou insira o conteúdo do arquivo na exibição de arquivo. Para conhecer um exemplo, veja a carga da solicitação de exemplo em [Importar partes de documentação com a API REST do API Gateway](#).
5. Escolha como lidar com os avisos na importação. Selecione Falhar nos avisos ou Ignorar avisos. Para obter mais informações, consulte [the section called “Erros e avisos ao importar a API para o API Gateway”](#).
6. Escolha Importar.

## Controlar o acesso à documentação da API no API Gateway

Se você possui uma equipe de documentação dedicada para escrever e editar a documentação da sua API, pode configurar permissões de acesso separadas para os seus desenvolvedores (para o desenvolvimento da API) e para os seus escritores ou editores (para o desenvolvimento de conteúdo). Isso é especialmente apropriado quando um fornecedor externo está envolvido na criação da documentação para você.

Para conceder acesso à sua equipe de documentação para criar, atualizar e publicar a documentação da API, você pode atribuir a ela uma função do IAM com a seguinte política do IAM, em que *account\_id* é o ID da conta da AWS da sua equipe de documentação.

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 obter informações sobre como definir permissões para acessar recursos do API Gateway, consulte [the section called “Como o Amazon API Gateway funciona com o IAM”](#).

## Gerar SDKs para APIs REST no API Gateway

Para chamar sua API REST em um modo específico de linguagem ou plataforma, você deve gerar o SDK específico de idioma ou de plataforma da API. Gere o SDK depois de criar, testar e implantar a API em um estágio. Atualmente, o API Gateway oferece suporte à geração de um SDK de uma API em Java, JavaScript, Java para Android, Objective-C ou Swift para iOS e Ruby.

Esta seção explica como gerar um SDK de uma API do API Gateway. Ela também demonstra como usar o SDK gerado em um aplicativo Java, um aplicativo Java para Android, aplicativos Objective-C e Swift para iOS e um aplicativo JavaScript.

Para facilitar a discussão, usamos esta [API](#) do API Gateway, que expõe a função de [Calculadora simples](#) do Lambda.

Antes de prosseguir, crie ou importe a API e implante-a pelo menos uma vez no API Gateway. Para instruções, consulte [Implantar APIs REST no API Gateway](#).

### Tópicos

- [Função de calculadora simples do Lambda](#)
- [API de calculadora simples no API Gateway](#)
- [Definição do OpenAPI da API de calculadora simples](#)
- [Gerar o SDK Java de uma API no API Gateway](#)
- [Gerar o SDK Android de uma API no API Gateway](#)
- [Gerar o SDK iOS de uma API no API Gateway](#)

- [Gerar o SDK JavaScript de uma API REST no API Gateway](#)
- [Gerar o SDK Ruby de uma API no API Gateway](#)
- [Gerar SDKs para uma API usando comandos da AWS CLI no API Gateway](#)

## Função de calculadora simples do Lambda

Como ilustração, usaremos uma função do Lambda em Node.js que realiza as operações binárias de adição, subtração, multiplicação e divisão.

### Tópicos

- [Formato de entrada da função de calculadora simples do Lambda](#)
- [Formato de saída da função de calculadora simples do Lambda](#)
- [Implementação da função de calculadora simples do Lambda](#)

### Formato de entrada da função de calculadora simples do Lambda

Essa função recebe uma entrada do seguinte formato:

```
{ "a": "Number", "b": "Number", "op": "string" }
```

em que op pode ser qualquer um de (+, -, \*, /, add, sub, mul, div).

### Formato de saída da função de calculadora simples do Lambda

Quando uma operação é bem-sucedida, ele retorna o resultado no seguinte formato:

```
{ "a": "Number", "b": "Number", "op": "string", "c": "Number" }
```

em que c guarda o resultado do cálculo.

### Implementação da função de calculadora simples do Lambda

A implementação da função do Lambda é a seguinte:

```
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";
 }

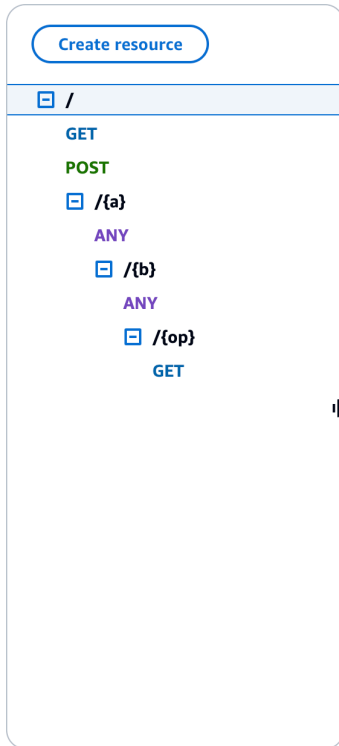
 const res = {};
 res.a = Number(event.a);
 res.b = Number(event.b);
 res.op = event.op;
 if (isNaN(event.a) || isNaN(event.b)) {
 return "400 Invalid Operand";
 }
 switch (event.op) {
 case "+":
 case "add":
 res.c = res.a + res.b;
 break;
 case "-":
 case "sub":
 res.c = res.a - res.b;
 break;
 case "*":
 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 simples no API Gateway

Nossa API de calculadora simples expõe três métodos (GET, POST, GET) para invocar o [the section called “Função de calculadora simples do Lambda”](#). Uma representação gráfica dessa API é mostrada da seguinte forma:

### Resources



Esses três métodos mostram maneiras diferentes de fornecer a entrada para a função do Lambda de backend para executar a mesma operação:

- O método GET `/?a=...&b=...&op=...` usa os parâmetros de consulta para especificar a entrada.
- O método POST `/` usa uma carga JSON de `{"a": "Number", "b": "Number", "op": "string"}` para especificar a entrada.
- O método GET `/{a}/{b}/{op}` usa os parâmetros de caminho para especificar a entrada.

Se não estiver definido, o API Gateway gerará o nome do método de SDK correspondente, combinando as partes de método e caminho HTTP. A parte do caminho raiz (`/`) é referida como `Api Root`. Por exemplo, o nome do método de SDK Java padrão para o método de API de GET `/?a=...&b=...&op=...` é `getABOp`, o nome do método de SDK padrão para POST `/` é



`postApiRoot`, e o nome do método de SDK padrão para GET `/a/b/op` é `getABOp`. SDKs individuais podem personalizar a convenção. Consulte a documentação na origem do SDK gerado para obter os nomes de método específicos do SDK.

Você pode, e deve, substituir os nomes de método de SDK padrão, especificando a propriedade `operationName` em cada método de API. Você pode fazer isso ao [criar o método de API](#) ou ao [atualizar o método de API](#) usando a API REST do API Gateway. Na definição do Swagger da API, você pode definir o `operationId` para obter o mesmo resultado.

Antes de mostrar como chamar esses métodos usando um SDK gerado pelo API Gateway para essa API, vamos lembrar brevemente como configurá-los. Para obter instruções detalhadas, consulte [Desenvolver APIs REST no API Gateway](#). Se você ainda é iniciante com o API Gateway, consulte [Escolher um tutorial de integração do AWS Lambda](#) primeiro.

### Criar modelos para entrada e saída

Para especificar uma entrada de tipo forte no SDK, criamos um modelo `Input` para a API: Para descrever o tipo de dados do corpo de resposta, criamos um modelo `Output` e um `Result`.

### Como criar modelos para entrada, saída e resultado

1. No painel de navegação principal, selecione Modelos.
2. Escolha Criar modelo.
3. Em Nome, digite **input**.
4. Em Tipo de conteúdo, insira **application/json**.

Se nenhum tipo de conteúdo correspondente for encontrado, a validação da solicitação não será executada. Para usar o mesmo modelo, independentemente do tipo de conteúdo, insira **\$default**.

5. Em Esquema do modelo, insira o seguinte 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. Escolha Criar modelo.
7. Repita as etapas a seguir para criar um modelo Output e um Result.

Para o modelo Output, insira o seguinte para o Esquema do modelo:

```
{
 "$schema": "http://json-schema.org/draft-04/schema#",
 "type": "object",
 "properties": {
 "c": {"type": "number"}
 },
 "title": "Output"
}
```

Para o modelo Result, insira o seguinte para o Esquema do modelo. Substitua o ID da API abc123 pelo ID de sua 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 parâmetros de consulta do método GET /

Para o método GET `/?a=..&b=..&op=..`, os parâmetros de consulta estão declarados em Method Request:

### Como configurar parâmetros de string de consulta GET/URL

1. Na seção Solicitação de método para o método GET no recurso de raiz (/), selecione Editar.

## 2. Selecione Parâmetros de string de consulta de URL e faça o seguinte:

- a. Escolha Add query string (Adicionar string de consulta).
- b. Em Nome, digite **a**.
- c. Mantenha Obrigatório e Armazenamento em cache desativados.
- d. Mantenha Armazenamento em cache desativado.

Repita as mesmas etapas e crie uma string de consulta chamada **b** e uma string de consulta chamada **op**.

## 3. Escolha Salvar.

Configurar o modelo de dados para a carga como entrada para o backend

Para o método POST `/`, criamos o modelo Input e o adicionamos à solicitação de método para definir a forma dos dados de entrada.

Como configurar o modelo de dados para a carga útil como entrada para o back-end

1. Na seção Solicitação de método para o método POST no recurso de raiz (`/`), selecione Editar.
2. Selecione Corpo da solicitação.
3. Escolha Add model (Adicionar modelo).
4. Em Tipo de conteúdo, insira **application/json**.
5. Em Modelo, selecione Entrada.
6. Escolha Salvar.

Com esse modelo, os clientes da sua API podem chamar o SDK para especificar a entrada, instanciando um objeto Input. Sem esse modelo, seus clientes precisariam criar o objeto de dicionário para representar a entrada JSON para a função do Lambda.

Configurar o modelo de dados para a saída do resultado do backend

Para todos os três métodos, criamos o modelo Result e o adicionamos ao Method Response do método para definir a forma da saída retornada pela função do Lambda.

Como configurar o modelo de dados para a saída do resultado do back-end

1. Selecione o recurso `/a/b/op` e o método GET.

2. Na guia Resposta de método, em Resposta 200, selecione Editar.
3. Em Corpo da resposta, selecione Adicionar modelo.
4. Em Tipo de conteúdo, insira **application/json**.
5. Em Modelo, selecione Resultado.
6. Escolha Salvar.

Com esse modelo, os clientes da sua API podem analisar uma saída bem-sucedida, lendo as propriedades de um objeto `Result`. Sem esse modelo, os clientes precisariam criar o objeto de dicionário para representar a saída JSON.

## Definição do OpenAPI da API de calculadora simples

Veja a seguir a definição do OpenAPI da API de calculadora simples. Você pode importá-lo para a sua conta. No entanto, você precisa redefinir as permissões baseadas em recurso na [função do Lambda](#) após a importação. Para fazer isso, volte a selecionar a função do Lambda criada na sua conta a partir da Integration Request (Solicitação de integração) no console do API Gateway. Isso fará com que o console do API Gateway redefina as permissões necessárias. Como alternativa, você pode usar o AWS Command Line Interface para o comando [add-permission](#) do 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\n \"a\" : $input.params('a'),\n \"b\" : $input.params('b'),\n \"op\" :\n \"$input.params('op')\"\n}\n",
 },
 "uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/\narn:aws:lambda:us-west-2:123456789012:function:Calc/invocations",
 "passthroughBehavior": "when_no_templates",
 "httpMethod": "POST",
 "responses": {
 "default": {
 "statusCode": "200",
 "responseTemplates": {

```

```

 "application/json": "#set($inputRoot = $input.path('$'))\n{\n
 \"input\" : {\n \"a\" : $inputRoot.a,\n \"b\" : $inputRoot.b,\n \"op\" :
 \"$inputRoot.op\"\n },\n \"output\" : {\n \"c\" : $inputRoot.c\n }\n}"
 }
 },
 "type": "aws"
}
},
"post": {
 "consumes": [
 "application/json"
],
 "produces": [
 "application/json"
],
 "parameters": [
 {
 "in": "body",
 "name": "Input",
 "required": true,
 "schema": {
 "$ref": "#/definitions/Input"
 }
 }
],
 "responses": {
 "200": {
 "description": "200 response",
 "schema": {
 "$ref": "#/definitions/Result"
 }
 }
 },
 "x-amazon-apigateway-integration": {
 "uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123456789012:function:Calc/invocations",
 "passthroughBehavior": "when_no_match",
 "httpMethod": "POST",
 "responses": {
 "default": {
 "statusCode": "200",
 "responseTemplates": {

```

```

 "application/json": "#set($inputRoot = $input.path('$'))\n{\n
 \"input\" : {\n \"a\" : $inputRoot.a,\n \"b\" : $inputRoot.b,\n \"op\" :
 \"$inputRoot.op\"\n },\n \"output\" : {\n \"c\" : $inputRoot.c\n }\n}"
 }
}
},
"type": "aws"
}
}
},
"/{a}": {
 "x-amazon-apigateway-any-method": {
 "consumes": [
 "application/json"
],
 "produces": [
 "application/json"
],
 "parameters": [
 {
 "name": "a",
 "in": "path",
 "required": true,
 "type": "string"
 }
],
 "responses": {
 "404": {
 "description": "404 response"
 }
 },
 "x-amazon-apigateway-integration": {
 "requestTemplates": {
 "application/json": "{\"statusCode\": 200}"
 },
 "passthroughBehavior": "when_no_match",
 "responses": {
 "default": {
 "statusCode": "404",
 "responseTemplates": {
 "application/json": "{ \"Message\" : \"Can't $context.httpMethod
$context.resourcePath\" }"
 }
 }
 }
 }
 }
}

```

```

 },
 "type": "mock"
 }
}
},
"/{a}/{b}": {
 "x-amazon-apigateway-any-method": {
 "consumes": [
 "application/json"
],
 "produces": [
 "application/json"
],
 "parameters": [
 {
 "name": "a",
 "in": "path",
 "required": true,
 "type": "string"
 },
 {
 "name": "b",
 "in": "path",
 "required": true,
 "type": "string"
 }
],
 "responses": {
 "404": {
 "description": "404 response"
 }
 },
 "x-amazon-apigateway-integration": {
 "requestTemplates": {
 "application/json": "{\"statusCode\": 200}"
 },
 "passthroughBehavior": "when_no_match",
 "responses": {
 "default": {
 "statusCode": "404",
 "responseTemplates": {
 "application/json": "{ \"Message\" : \"Can't $context.httpMethod $context.resourcePath\" }"
 }
 }
 }
 }
 }
}

```



```

 }
 },
 "type": "mock"
 }
 },
 "{a}/{b}/{op}": {
 "get": {
 "consumes": [
 "application/json"
],
 "produces": [
 "application/json"
],
 "parameters": [
 {
 "name": "a",
 "in": "path",
 "required": true,
 "type": "string"
 },
 {
 "name": "b",
 "in": "path",
 "required": true,
 "type": "string"
 },
 {
 "name": "op",
 "in": "path",
 "required": true,
 "type": "string"
 }
],
 "responses": {
 "200": {
 "description": "200 response",
 "schema": {
 "$ref": "#/definitions/Result"
 }
 }
 }
 },
 "x-amazon-apigateway-integration": {
 "requestTemplates": {

```

```

 "application/json": "#set($inputRoot = $input.path('$'))\n{\n
 \"a\" : $input.params('a'),\n \"b\" : $input.params('b'),\n \"op\" :
 \"$input.params('op')\"\n}"
 },
 "uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:123456789012:function:Calc/invocations",
 "passthroughBehavior": "when_no_templates",
 "httpMethod": "POST",
 "responses": {
 "default": {
 "statusCode": "200",
 "responseTemplates": {
 "application/json": "#set($inputRoot = $input.path('$'))\n{\n
 \"input\" : {\n \"a\" : $inputRoot.a,\n \"b\" : $inputRoot.b,\n \"op\" :
 \"$inputRoot.op\"\n },\n \"output\" : {\n \"c\" : $inputRoot.c\n }\n}"
 }
 }
 },
 "type": "aws"
 }
 }
},
"definitions": {
 "Input": {
 "type": "object",
 "properties": {
 "a": {
 "type": "number"
 },
 "b": {
 "type": "number"
 },
 "op": {
 "type": "string"
 }
 },
 "title": "Input"
 },
 "Output": {
 "type": "object",
 "properties": {
 "c": {
 "type": "number"
 }
 }
 }
}

```

```

 }
 },
 "title": "Output"
},
"Result": {
 "type": "object",
 "properties": {
 "input": {
 "$ref": "#/definitions/Input"
 },
 "output": {
 "$ref": "#/definitions/Output"
 }
 },
 "title": "Result"
}
}
}

```

## OpenAPI 3.0

```

{
 "openapi" : "3.0.1",
 "info" : {
 "title" : "SimpleCalc",
 "version" : "2016-09-29T20:27:30Z"
 },
 "servers" : [{
 "url" : "https://t6dve4zn25.execute-api.us-west-2.amazonaws.com/{basePath}",
 "variables" : {
 "basePath" : {
 "default" : "demo"
 }
 }
 }],
 "paths" : {
 ("/{a}/{b}" : {
 "x-amazon-apigateway-any-method" : {
 "parameters" : [{
 "name" : "a",
 "in" : "path",
 "required" : true,
 "schema" : {

```

```

 "type" : "string"
 }
 }, {
 "name" : "b",
 "in" : "path",
 "required" : true,
 "schema" : {
 "type" : "string"
 }
 }],
 "responses" : {
 "404" : {
 "description" : "404 response",
 "content" : { }
 }
 },
 "x-amazon-apigateway-integration" : {
 "type" : "mock",
 "responses" : {
 "default" : {
 "statusCode" : "404",
 "responseTemplates" : {
 "application/json" : "{ \"Message\" : \"Can't $context.httpMethod
$context.resourcePath\" }"
 }
 }
 },
 "requestTemplates" : {
 "application/json" : "{ \"statusCode\" : 200}"
 },
 "passthroughBehavior" : "when_no_match"
 }
 }
}, {
 "/{a}/{b}/{op}" : {
 "get" : {
 "parameters" : [{
 "name" : "a",
 "in" : "path",
 "required" : true,
 "schema" : {
 "type" : "string"
 }
 }
], {

```

```

 "name" : "b",
 "in" : "path",
 "required" : true,
 "schema" : {
 "type" : "string"
 }
 }, {
 "name" : "op",
 "in" : "path",
 "required" : true,
 "schema" : {
 "type" : "string"
 }
 }],
 "responses" : {
 "200" : {
 "description" : "200 response",
 "content" : {
 "application/json" : {
 "schema" : {
 "$ref" : "#/components/schemas/Result"
 }
 }
 }
 }
 },
 "x-amazon-apigateway-integration" : {
 "type" : "aws",
 "httpMethod" : "POST",
 "uri" : "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:111122223333:function:Calc/invocations",
 "responses" : {
 "default" : {
 "statusCode" : "200",
 "responseTemplates" : {
 "application/json" : "#set($inputRoot = $input.path('$'))\n{\n
\\\"input\\\" : {\n \\\"a\\\" : $inputRoot.a,\n \\\"b\\\" : $inputRoot.b,\n \\\"op\\\" :
\\\"$inputRoot.op\\\" }\n },\n \\\"output\\\" : {\n \\\"c\\\" : $inputRoot.c\n }\n}"
 }
 }
 },
 "requestTemplates" : {

```

```

 "application/json" : "#set($inputRoot = $input.path('$'))\n{\n
 \"a\" : $input.params('a'),\n \"b\" : $input.params('b'),\n \"op\" :
 \"$input.params('op')\"\n}"
 },
 "passthroughBehavior" : "when_no_templates"
 }
 },
 {
 "/": {
 "get": {
 "parameters": [{
 "name" : "op",
 "in" : "query",
 "schema": {
 "type" : "string"
 }
 }, {
 "name" : "a",
 "in" : "query",
 "schema": {
 "type" : "string"
 }
 }, {
 "name" : "b",
 "in" : "query",
 "schema": {
 "type" : "string"
 }
 }
],
 "responses": {
 "200": {
 "description" : "200 response",
 "content": {
 "application/json": {
 "schema": {
 "$ref" : "#/components/schemas/Result"
 }
 }
 }
 }
 }
 },
 "x-amazon-apigateway-integration": {
 "type" : "aws",
 "httpMethod" : "POST",

```

```

 "uri" : "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:111122223333:function:Calc/invocations",
 "responses" : {
 "default" : {
 "statusCode" : "200",
 "responseTemplates" : {
 "application/json" : "#set($inputRoot = $input.path('$'))\n{\n
\n\"input\" : {\n \"a\" : $inputRoot.a,\n \"b\" : $inputRoot.b,\n \"op\" :
\n\"$inputRoot.op\"\n },\n \"output\" : {\n \"c\" : $inputRoot.c\n }\n}"
 }
 }
 },
 "requestTemplates" : {
 "application/json" : "#set($inputRoot = $input.path('$'))\n{\n
\n\"a\" : $input.params('a'),\n \"b\" : $input.params('b'),\n \"op\" :
\n\"$input.params('op')\"\n}"
 },
 "passthroughBehavior" : "when_no_templates"
}
},
"post" : {
 "requestBody" : {
 "content" : {
 "application/json" : {
 "schema" : {
 "$ref" : "#/components/schemas/Input"
 }
 }
 }
 },
 "required" : true
},
"responses" : {
 "200" : {
 "description" : "200 response",
 "content" : {
 "application/json" : {
 "schema" : {
 "$ref" : "#/components/schemas/Result"
 }
 }
 }
 }
}
},
"x-amazon-apigateway-integration" : {

```

```

 "type" : "aws",
 "httpMethod" : "POST",
 "uri" : "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-west-2:111122223333:function:Calc/invocations",
 "responses" : {
 "default" : {
 "statusCode" : "200",
 "responseTemplates" : {
 "application/json" : "#set($inputRoot = $input.path('$'))\n{\n
\n\"input\" : {\n \"a\" : $inputRoot.a,\n \"b\" : $inputRoot.b,\n \"op\" :
\n\"$inputRoot.op\"\n },\n \"output\" : {\n \"c\" : $inputRoot.c\n }\n}"
 }
 }
 },
 "passthroughBehavior" : "when_no_match"
 }
},
"/{a}" : {
 "x-amazon-apigateway-any-method" : {
 "parameters" : [{
 "name" : "a",
 "in" : "path",
 "required" : true,
 "schema" : {
 "type" : "string"
 }
 }],
 "responses" : {
 "404" : {
 "description" : "404 response",
 "content" : { }
 }
 },
 "x-amazon-apigateway-integration" : {
 "type" : "mock",
 "responses" : {
 "default" : {
 "statusCode" : "404",
 "responseTemplates" : {
 "application/json" : "{ \"Message\" : \"Can't $context.httpMethod
$context.resourcePath\" }"
 }
 }
 }
 }
 }
}

```



```
 },
 "requestTemplates" : {
 "application/json" : "{\"statusCode\": 200}"
 },
 "passthroughBehavior" : "when_no_match"
 }
}
},
"components" : {
 "schemas" : {
 "Input" : {
 "title" : "Input",
 "type" : "object",
 "properties" : {
 "a" : {
 "type" : "number"
 },
 "b" : {
 "type" : "number"
 },
 "op" : {
 "type" : "string"
 }
 }
 },
 "Output" : {
 "title" : "Output",
 "type" : "object",
 "properties" : {
 "c" : {
 "type" : "number"
 }
 }
 }
 },
 "Result" : {
 "title" : "Result",
 "type" : "object",
 "properties" : {
 "input" : {
 "$ref" : "#/components/schemas/Input"
 },
 "output" : {
 "$ref" : "#/components/schemas/Output"
 }
 }
 }
}
```

```
}
}
}
}
}
}
```

## Gerar o SDK Java de uma API no API Gateway

O procedimento a seguir mostra como gerar o SDK Java de uma API no API Gateway.

Como gerar o SDK do Java de uma API no API Gateway

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST.
3. Escolha Stages (Estágios).
4. No painel Estágios, escolha o nome do estágio.
5. Abra o menu Ações do estágio e escolha Gerar SDK.
6. Em Plataforma, escolha a plataforma Java e faça o seguinte:
  - a. Para Service Name, especifique o nome do seu SDK. Por exemplo, **SimpleCalcSdk**. Ele se tornará o nome da sua classe de cliente SDK. O nome corresponde à tag <name> em <project>, no arquivo pom.xml, que está na pasta de projeto do SDK. Não inclua hifens.
  - b. Para Java Package Name, especifique um nome de pacote para o seu SDK. Por exemplo, **examples.aws.apig.simpleCalc.sdk**. Esse nome de pacote é usado como o namespace da sua biblioteca SDK. Não inclua hifens.
  - c. Em Java Build System (Sistema de compilação Java), insira **maven** ou **gradle** para especificar o sistema de compilação.
  - d. Em Java Group Id (Id de grupo Java), insira um identificador de grupo para seu projeto SDK. Por exemplo, digite **my-apig-api-examples**. Esse identificador corresponde à tag <groupId> em <project>, no arquivo pom.xml, que está na pasta de projeto do SDK.
  - e. Em Java Artifact Id (Id de artefato Java), insira um identificador de artefato para o projeto de SDK. Por exemplo, digite **simple-calc-sdk**. Esse identificador corresponde à tag <artifactId> em <project>, no arquivo pom.xml, que está na pasta de projeto do SDK.

- f. Para Java Artifact Version (Versão do artefato Java), insira uma string de identificador de versão. Por exemplo, **1.0.0**. Esse identificador de versão corresponde à tag `<version>` em `<project>`, no arquivo `pom.xml`, que está na pasta de projeto do SDK.
  - g. Em Source Code License Text (Texto de licença de código de origem), insira o texto da licença do código fonte, se houver.
7. Escolha Generate SDK (Gerar SDK), e siga as instruções na tela para fazer download do SDK gerado pelo API Gateway.

Siga as instruções em [Usar um SDK Java gerado pelo API Gateway para uma API REST](#) para usar o SDK gerado.

Toda vez que você atualiza uma API, deve reimplantá-la e gerar novamente o SDK para incluir as atualizações.

## Gerar o SDK Android de uma API no API Gateway

O procedimento a seguir mostra como gerar o SDK Android de uma API no API Gateway.

Como gerar o SDK do Android de uma API no API Gateway

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST.
3. Escolha Stages (Estágios).
4. No painel Estágios, escolha o nome do estágio.
5. Abra o menu Ações do estágio e escolha Gerar SDK.
6. Em Plataforma, escolha a plataforma Android e faça o seguinte:
  - a. Em Group ID (ID do grupo), insira o identificador exclusivo do projeto correspondente. Ele é usado no arquivo `pom.xml` (por exemplo, **com.mycompany**).
  - b. Em Invoker package (Pacote do invocador), insira o namespace das classes de cliente geradas (por exemplo, **com.mycompany.clientsdk**).
  - c. Em Artifact ID (ID do artefato), insira o nome do arquivo `.jar` compilado sem a versão. Ele é usado no arquivo `pom.xml` (por exemplo, **aws-apigateway-api-sdk**).
  - d. Em Artifact version (Versão do artefato), insira o número da versão do artefato do cliente gerado. Ele é usado no arquivo `pom.xml` e deve seguir um padrão *principal.secundário.patch* (por exemplo, **1.0.0**).

7. Escolha Generate SDK (Gerar SDK), e siga as instruções na tela para fazer download do SDK gerado pelo API Gateway.

Siga as instruções em [Usar um SDK Android gerado pelo API Gateway para uma API REST](#) para usar o SDK gerado.

Toda vez que você atualiza uma API, deve reimplantá-la e gerar novamente o SDK para incluir as atualizações.

## Gerar o SDK iOS de uma API no API Gateway

O procedimento a seguir mostra como gerar o SDK iOS de uma API no API Gateway.

Como gerar o SDK do iOS de uma API no API Gateway

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST.
3. Escolha Stages (Estágios).
4. No painel Estágios, escolha o nome do estágio.
5. Abra o menu Ações do estágio e escolha Gerar SDK.
6. Em Plataforma, escolha a plataforma iOS (Objective-C) ou iOS (Swift) e faça o seguinte:
  - Digite um prefixo exclusivo na caixa Prefix.

O efeito do prefixo é o seguinte: se você atribuir, por exemplo, **SIMPLE\_CALC** como o prefixo para o SDK da API [SimpleCalc](#) com os modelos input, output e result, o SDK gerado conterá a classe `SIMPLE_CALCSimpleCalcClient` que encapsula a API, incluindo as solicitações/respostas do método. Além disso, o SDK gerado conterá as classes `SIMPLE_CALCinput`, `SIMPLE_CALCoutput` e `SIMPLE_CALCresult` para representar a entrada, a saída e os resultados, respectivamente, para representar a entrada de solicitação e a saída da resposta. Para obter mais informações, consulte [Usar o SDK iOS gerado pelo API Gateway para uma API REST em Objective-C ou Swift](#).

7. Escolha Generate SDK (Gerar SDK), e siga as instruções na tela para fazer download do SDK gerado pelo API Gateway.

Siga as instruções em [Usar o SDK iOS gerado pelo API Gateway para uma API REST em Objective-C ou Swift](#) para usar o SDK gerado.

Toda vez que você atualiza uma API, deve reimplantá-la e gerar novamente o SDK para incluir as atualizações.

## Gerar o SDK JavaScript de uma API REST no API Gateway

O procedimento a seguir mostra como gerar o SDK JavaScript de uma API no API Gateway.

Como gerar o SDK do JavaScript de uma API no API Gateway

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST.
3. Escolha Stages (Estágios).
4. No painel Estágios, escolha o nome do estágio.
5. Abra o menu Ações do estágio e escolha Gerar SDK.
6. Em Plataforma, escolha a plataforma JavaScript.
7. Escolha Generate SDK (Gerar SDK), e siga as instruções na tela para fazer download do SDK gerado pelo API Gateway.

Siga as instruções em [Usar um SDK JavaScript gerado pelo API Gateway para uma API REST](#) para usar o SDK gerado.

Toda vez que você atualiza uma API, deve reimplantá-la e gerar novamente o SDK para incluir as atualizações.

## Gerar o SDK Ruby de uma API no API Gateway

O procedimento a seguir mostra como gerar o SDK Ruby de uma API no API Gateway.

Como gerar o SDK do Ruby de uma API no API Gateway

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST.
3. Escolha Stages (Estágios).
4. No painel Estágios, escolha o nome do estágio.
5. Abra o menu Ações do estágio e escolha Gerar SDK.
6. Em Plataforma, escolha a plataforma Ruby e faça o seguinte:

- a. Para Service Name, especifique o nome do seu SDK. Por exemplo, **SimpleCalc**. Isso é usado para gerar o namespace Ruby Gem de sua API. O nome deve conter somente letras (a-zA-Z), sem nenhum outro caractere especial ou número.
  - b. Em Ruby Gem Name, especifique o nome do Ruby Gem que receberá o código-fonte do SDK gerado para sua API. Por padrão, é o nome do serviço em minúsculas, acrescido do sufixo `-sdk`; por exemplo, **simplecalc-sdk**.
  - c. Em Ruby Gem Version, especifique um número de versão para o Ruby Gem gerado. Por padrão, ele é definido como `1.0.0`.
7. Escolha Generate SDK (Gerar SDK), e siga as instruções na tela para fazer download do SDK gerado pelo API Gateway.

Siga as instruções em [Usar um SDK Ruby gerado pelo API Gateway para uma API REST](#) para usar o SDK gerado.

Toda vez que você atualiza uma API, deve reimplantá-la e gerar novamente o SDK para incluir as atualizações.

## Gerar SDKs para uma API usando comandos da AWS CLI no API Gateway

Você pode usar a AWS CLI para gerar e fazer download do SDK de uma API para uma plataforma compatível usando o comando [get-sdk](#). Demonstramos isso para algumas das plataformas compatíveis a seguir.

### Tópicos

- [Gerar e fazer download do SDK do Java para Android usando a AWS CLI](#)
- [Gerar e fazer download do SDK do JavaScript usando a AWS CLI](#)
- [Gerar e fazer download do SDK do Ruby usando a AWS CLI](#)

### Gerar e fazer download do SDK do Java para Android usando a AWS CLI

Para gerar e fazer download de um SDK do Java para Android gerado pelo API Gateway de uma API (udpuvzbkc) em determinado estágio (test), chame o comando da seguinte forma:

```
aws apigateway get-sdk \
 --rest-api-id udpuvzbkc \
 --stage-name test \
 --output-type json
```

```
--sdk-type android \
--parameters groupId='com.mycompany',\
 invokerPackage='com.mycompany.myApiSdk',\
 artifactId='myApiSdk',\
 artifactVersion='0.0.1' \
~/apps/myApi/myApi-android-sdk.zip
```

A última entrada do `~/apps/myApi/myApi-android-sdk.zip` é o caminho para o arquivo de SDK obtido por download chamado `myApi-android-sdk.zip`.

Gerar e fazer download do SDK do JavaScript usando a AWS CLI

Para gerar e fazer download do SDK do JavaScript gerado pelo API Gateway de uma API (`udpuvvzbkc`) em determinado estágio (`test`), chame o comando da seguinte forma:

```
aws apigateway get-sdk \
 --rest-api-id udpuvvzbkc \
 --stage-name test \
 --sdk-type javascript \
 ~/apps/myApi/myApi-js-sdk.zip
```

A última entrada do `~/apps/myApi/myApi-js-sdk.zip` é o caminho para o arquivo de SDK obtido por download chamado `myApi-js-sdk.zip`.

Gerar e fazer download do SDK do Ruby usando a AWS CLI

Para gerar e fazer download do SDK do Ruby de uma API (`udpuvvzbkc`) em um determinado estágio (`test`), use o comando da seguinte forma:

```
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
```

A última entrada do `~/apps/myApi/myApi-ruby-sdk.zip` é o caminho para o arquivo de SDK obtido por download chamado `myApi-ruby-sdk.zip`.

Em seguida, mostramos como usar o SDK gerado para chamar a API subjacente. Para obter mais informações, consulte [Invocar APIs REST no API Gateway](#).

## Venda suas APIs do API Gateway pelo AWS Marketplace

Depois de criar, testar e implantar suas APIs, você poderá empacotá-las em um [plano de uso](#) do API Gateway e vender o plano como um produto de Software como Serviço (SaaS) via AWS Marketplace. Os compradores da API que se inscrevem na sua oferta de produtos são cobrados pelo AWS Marketplace com base no número de solicitações feitas no plano de uso.

Para vender suas APIs no AWS Marketplace, você deve configurar o canal de vendas para integrar o AWS Marketplace ao API Gateway. De modo geral, isso envolve listar seu produto no AWS Marketplace, configurar uma função do IAM com políticas apropriadas para permitir que o API Gateway envie métricas de uso para o AWS Marketplace, associar um produto do AWS Marketplace a um plano de uso do API Gateway e associar um comprador do AWS Marketplace a uma chave de API do API Gateway. Os detalhes são discutidos nas seções a seguir.

Para obter mais informações sobre como vender sua API como um produto de SaaS no AWS Marketplace, consulte o [Guia do usuário do AWS Marketplace](#).

### Tópicos

- [Inicializar a integração do AWS Marketplace com o API Gateway](#)
- [Gerenciar a assinatura do cliente para planos de uso](#)

## Inicializar a integração do AWS Marketplace com o API Gateway

As tarefas a seguir destinam-se à inicialização uma única vez da integração do AWS Marketplace com o API Gateway, o que permite vender suas APIs como um produto de SaaS.

### Listar um produto no AWS Marketplace

Para listar seu plano de uso como um produto SaaS, envie um formulário de carregamento de produto pelo [AWS Marketplace](#). O produto deve conter uma dimensão denominada `apigateway` do tipo `requests`. Essa dimensão define o preço por solicitação e é usada pelo API Gateway para medir solicitações para suas APIs.

### Criar a função de medição

Crie uma função do IAM denominada `ApiGatewayMarketplaceMeteringRole` com a seguinte política de execução e política de confiança. Essa função permite que o API Gateway envie métricas de uso para o AWS Marketplace em seu nome.



## Política de execução da função de medição

### JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Action": [
 "aws-marketplace:BatchMeterUsage",
 "aws-marketplace:ResolveCustomer"
],
 "Resource": "*",
 "Effect": "Allow"
 }
]
}
```

## Política de relacionamento confiável da função de medição

### JSON

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

## Associar o plano de uso com um produto do AWS Marketplace

Ao listar um produto no AWS Marketplace, você recebe um código de produto do AWS Marketplace. Para integrar o API Gateway com o AWS Marketplace, associe seu plano de uso ao código

de produto do AWS Marketplace. Habilite a associação definindo o campo [productCode](#) do UsagePlan do API Gateway como o código do seu produto do AWS Marketplace e usando o console do API Gateway, a API REST do API Gateway, a AWS CLI para o API Gateway ou o SDK da AWS para o API Gateway. O exemplo de código a seguir usa a API REST do API Gateway:

```
PATCH /usageplans/USAGE_PLAN_ID
Host: apigateway.region.amazonaws.com
Authorization: ...

{
 "patchOperations" : [{
 "path" : "/productCode",
 "value" : "MARKETPLACE_PRODUCT_CODE",
 "op" : "replace"
 }]
}
```

## Gerenciar a assinatura do cliente para planos de uso

As seguintes tarefas são manipuladas pelo aplicativo do portal do desenvolvedor.

Quando um cliente se inscreve no seu produto via AWS Marketplace, o AWS Marketplace encaminha uma solicitação POST para o URL de assinaturas de SaaS que você registrou ao publicar seu produto no AWS Marketplace. A solicitação POST acompanha um parâmetro `x-amzn-marketplace-token` que contém informações do comprador. Siga as instruções na [integração de clientes de SaaS](#) para lidar com esse redirecionamento na aplicação do portal do desenvolvedor.

Respondendo à solicitação de inscrição de um cliente, o AWS Marketplace envia uma notificação `subscribe-success` para um tópico do Amazon SNS no qual você pode se inscrever. (Consulte [Integração de clientes de SaaS](#)). Para aceitar a solicitação de assinatura do cliente, você lida com a notificação `subscribe-success` criando ou recuperando uma chave de API do API Gateway para o cliente, associando o `customerId` provisionado pelo AWS Marketplace do cliente às chaves da API e, em seguida, adicionando a chave da API ao seu plano de uso.

Quando a solicitação de assinatura do cliente for concluída, o aplicativo de portal do desenvolvedor deverá apresentar ao cliente a chave de API associada e informá-lo de que essa chave deverá ser incluída no cabeçalho `x-api-key` em solicitações para as APIs.

Quando um cliente cancela uma assinatura de um plano de uso, o AWS Marketplace envia uma notificação `unsubscribe-success` ao tópico do SNS. Para concluir o processo de cancelamento

da assinatura do cliente, você pode manipular a notificação `unsubscribe-success` removendo chaves de API do cliente do plano de uso.

### Autorizar um cliente a acessar um plano de uso

Para autorizar o acesso ao seu plano de uso para um cliente, use a API do API Gateway para buscar ou criar uma chave de API para o cliente e adicione essa chave de API ao plano de uso.

O exemplo a seguir mostra como chamar a API REST do API Gateway para criar uma nova chave de API com um valor de `customerId` específico do AWS Marketplace (*`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"
}
```

O exemplo a seguir mostra como obter uma chave de API com um valor AWS Marketplace específico do `customerId` (*`MARKETPLACE_CUSTOMER_ID`*).

```
GET apikeys?customerId=MARKETPLACE_CUSTOMER_ID HTTP/1.1
Host: apigateway.region.amazonaws.com
Authorization: ...
```

Para adicionar uma chave de API a um plano de uso, crie uma [UsagePlanKey](#) com a chave de API para o plano de uso relevante. O exemplo a seguir mostra como fazer isso usando a API REST do API Gateway, em que `n371pt` é o ID do plano de uso e `q5ugs7qjjh` é um `keyId` de API de exemplo retornado dos exemplos anteriores.

```
POST /usageplans/n371pt/keys HTTP/1.1
```

```
Host: apigateway.region.amazonaws.com
Authorization: ...

{
 "keyId": "q5ugs7qjjh",
 "keyType": "API_KEY"
}
```

## Associar um cliente a uma chave de API

É necessário atualizar o campo `customerId` de [ApiKey](#) para o ID de cliente do AWS Marketplace. Isto associa a chave de API ao cliente do AWS Marketplace, o que permite a medição e o faturamento para o comprador. O exemplo de código a seguir chama a API REST do API Gateway para fazer isso.

```
PATCH /apikeys/q5ugs7qjjh
Host: apigateway.region.amazonaws.com
Authorization: ...

{
 "patchOperations" : [{
 "path" : "/customerId",
 "value" : "MARKETPLACE_CUSTOMER_ID",
 "op" : "replace"
 }]
}
```

## Proteger as APIs REST no API Gateway

O API Gateway fornece várias maneiras de proteger sua API de determinadas ameaças, como usuários mal-intencionados ou picos de tráfego. Você pode proteger a sua API usando estratégias, como gerar certificados SSL, configurar um firewall de aplicação Web, definir alvos de controle de utilização e permitir apenas o acesso à sua API de uma Virtual Private Cloud (VPC). Nesta seção, você pode aprender a habilitar esses recursos usando o API Gateway.

### Tópicos

- [Como ativar a autenticação do TLS mútuo para APIs REST no API Gateway](#)
- [Gerar e configurar um certificado SSL para autenticação de backend no API Gateway](#)
- [Usar o AWS WAF para proteger as APIs REST no API Gateway](#)

- [Controle de utilização das solicitações das APIs REST para ter um melhor throughput no API Gateway](#)
- [APIs REST privadas no API Gateway](#)

## Como ativar a autenticação do TLS mútuo para APIs REST no API Gateway

A autenticação TLS mútua requer autenticação bidirecional entre o cliente e o servidor. Com TLS mútuo, os clientes devem apresentar certificados X.509 para verificar sua identidade a fim de acessar sua API. O TLS mútuo é um requisito comum para a Internet das Coisas (IoT) e aplicações business-to-business.

É possível usar o TLS mútuo juntamente com outras [operações de autorização e autenticação](#) compatíveis com o API Gateway. O API Gateway encaminha os certificados que os clientes fornecem aos autorizadores do Lambda e às integrações de backend.

### Important

Por padrão, os clientes podem invocar sua API usando o endpoint `execute-api` gerado pelo API Gateway para sua API. Para garantir que os clientes possam acessar sua API somente usando um nome de domínio personalizado com TLS mútuo, desabilite o endpoint `execute-api` padrão. Para saber mais, consulte [the section called “Desabilitar o endpoint padrão”](#).

### Tópicos

- [Pré-requisitos do TLS mútuo](#)
- [Configurar o TLS mútuo para um nome de domínio personalizado](#)
- [Invocar uma API usando um nome de domínio personalizado que requer TLS mútuo](#)
- [Atualizar o armazenamento de confiança](#)
- [Desativar o TLS mútuo](#)
- [Solucionar problemas de TLS mútuo para a API REST](#)

## Pré-requisitos do TLS mútuo

Para configurar o TLS mútuo, você precisa de:

- Um nome de domínio regional personalizado
- Pelo menos um certificado configurado no AWS Certificate Manager para o seu nome de domínio personalizado
- Um armazenamento de confiança configurado e carregado no Amazon S3

## Nomes de domínios personalizados

Para habilitar o TLS mútuo para uma API REST, é necessário configurar um nome de domínio personalizado para sua API. Você pode habilitar o TLS mútuo para um nome de domínio personalizado e, depois, fornecer o nome de domínio personalizado aos clientes. Para acessar uma API usando um nome de domínio personalizado que tenha TLS mútuo habilitado, os clientes devem apresentar certificados confiáveis em solicitações de API. Você pode encontrar essas informações em [the section called “Nomes de domínios personalizados”](#).

## Uso de certificados emitidos pelo AWS Certificate Manager

É possível solicitar um certificado publicamente confiável diretamente do ACM ou importar certificados públicos ou autoassinados. Para configurar um certificado no ACM, acesse o [ACM](#). Para importar um certificado, continue lendo na seção a seguir.

## Usar um certificado importado ou Autoridade de Certificação Privada da AWS

Para usar um certificado importado para o ACM ou um certificado da Autoridade de Certificação Privada da AWS com TLS mútuo, o API Gateway precisa de um `ownershipVerificationCertificate` emitido pela ACM. Esse certificado de propriedade é utilizado apenas para verificar você se tem permissões para utilizar o nome de domínio. Ele não é usado para o handshake TLS. Se você ainda não tem um `ownershipVerificationCertificate`, acesse <https://console.aws.amazon.com/acm/> para configurar um.

Você precisará manter esse certificado válido durante todo o tempo de vida do seu nome de domínio. Se um certificado expirar e a renovação automática falhar, todas as atualizações do nome de domínio serão bloqueadas. Você precisará atualizar o `ownershipVerificationCertificateArn` com um `ownershipVerificationCertificate` válido antes de poder fazer outras alterações. O `ownershipVerificationCertificate` não pode ser usado como um certificado de servidor para outro domínio de TLS mútuo no API Gateway. Se um certificado for reimportado diretamente para o ACM, o emissor deverá permanecer o mesmo.

## Configuração do armazenamento de confiança

Os armazenamentos de confiança são arquivos de texto com extensão `.pem`. Eles são uma lista confiável de certificados de autoridades de certificação. Para usar TLS mútuo, crie um armazenamento confiável de certificados X.509 que podem acessar sua API.

Você deve incluir a cadeia de confiança completa, começando pelo certificado da autoridade de certificação emissora até o certificado CA, em seu armazenamento de confiança. O API Gateway aceita certificados de cliente emitidos por qualquer autoridade de certificação presente na cadeia de confiança. Os certificados podem ser de autoridades de certificação públicas ou privadas. Eles podem ter um tamanho máximo de cadeia de quatro. Você também pode fornecer certificados autoassinados. Os seguintes algoritmos são aceitos no armazenamento de confiança:

- SHA-256 ou mais forte
- RSA-2048 ou mais forte
- ECDSA-256 ou ECDSA-384

O API Gateway valida várias propriedades de certificado. É possível usar autorizadores do Lambda para executar verificações adicionais quando um cliente invoca uma API, incluindo verificar se um certificado foi revogado. O API Gateway valida as seguintes propriedades:

| Validação                                    | Descrição                                                                                                                           |
|----------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| Sintaxe X.509                                | O certificado deve atender aos requisitos da sintaxe X.509.                                                                         |
| Integridade                                  | O conteúdo do certificado não pode ter sido alterado do assinado pela autoridade de certificação do armazenamento confiável.        |
| Validity                                     | O período de validade do certificado deve ser atual.                                                                                |
| Encadeamento de nomes/encadeamento de chaves | Os nomes e os assuntos dos certificados devem formar uma cadeia ininterrupta. Eles podem ter um tamanho máximo de cadeia de quatro. |

Fazer upload do armazenamento de confiança para um bucket do Amazon S3 em um único arquivo

Veja a seguir um exemplo da possível aparência de um arquivo .pem.

Example certificates.pem

```
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
...
```

O comando `cp` da AWS CLI indicado abaixo carrega `certificates.pem` em seu bucket do Amazon S3.

```
aws s3 cp certificates.pem s3://bucket-name
```

## Configurar o TLS mútuo para um nome de domínio personalizado

Para configurar o TLS mútuo para uma API REST, é necessário usar um nome de domínio personalizado regional para sua API, com uma política de segurança TLS\_1\_2. Para saber mais sobre como escolher uma política de segurança, consulte [the section called “Escolher uma política de segurança”](#).

### Note

O TLS mútuo não é suportado para APIs privadas.

Depois de fazer upload do armazenamento de confiança para o Amazon S3, você pode configurar o nome de domínio personalizado para usar TLS mútuo. O comando `create-domain-name` indicado abaixo cria um nome de domínio personalizado com TLS mútuo:

```
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
```

Depois de criar o nome de domínio, é necessário configurar registros DNS e mapeamentos de caminho base para operações da API. Para saber mais, consulte [Configurar um nome de domínio regional personalizado no API Gateway](#).

## Invocar uma API usando um nome de domínio personalizado que requer TLS mútuo

Para invocar uma API com TLS mútuo habilitado, os clientes precisam apresentar um certificado confiável na solicitação de API. Quando um cliente tenta invocar a API, o API Gateway procura o emissor do certificado de cliente no seu armazenamento de confiança. Para que o API Gateway prossiga com a solicitação, o emissor do certificado e a cadeia de confiança completa até o certificado CA raiz devem estar no seu armazenamento de confiança.

O comando `curl` demonstrativo a seguir envia uma solicitação para `api.example.com`, que inclui `my-cert.pem` na solicitação. `my-key.key` é a chave privada para o certificado.

```
curl -v --key ./my-key.key --cert ./my-cert.pem api.example.com
```

Sua API será invocada somente se o seu armazenamento de confiança confiar no certificado. As seguintes condições farão com que o API Gateway falhe no handshake TLS e negue a solicitação com um código de status 403. Se seu certificado:

- não é confiável
- expirou
- não usa um algoritmo compatível

### Note

O API Gateway não verifica se um certificado foi revogado.

## Atualizar o armazenamento de confiança

Para atualizar os certificados no armazenamento de confiança, faça upload de um novo pacote de certificados para o Amazon S3. Em seguida, você poderá atualizar o nome de domínio personalizado para usar o certificado atualizado.

Use o [Versionamento do Amazon S3](#) para manter várias versões do armazenamento de confiança. Quando o nome de domínio personalizado é atualizado para usar uma nova versão de armazenamento de confiança, o API Gateway exibirá avisos se os certificados forem inválidos.

O API Gateway produz avisos de certificado somente quando o nome de domínio é atualizado. O API Gateway não o notificará se um certificado carregado anteriormente expirar.

O comando [update-domain-name](#) indicado abaixo atualiza um nome de domínio personalizado para usar uma nova versão do armazenamento de confiança:

```
aws apigateway update-domain-name \
 --domain-name api.example.com \
 --patch-operations op='replace',path='/mutualTlsAuthentication/
truststoreVersion',value='abcdef123'
```

## Desativar o TLS mútuo

O comando [update-domain-name](#) indicado abaixo desabilita o TLS mútuo:

```
aws apigateway update-domain-name \
 --domain-name api.example.com \
 --patch-operations op='replace',path='/mutualTlsAuthentication/
truststoreUri',value=''
```

## Solucionar problemas de TLS mútuo para a API REST

O tópico a seguir fornece orientações para a solução de erros e problemas que você pode encontrar ao ativar o TLS mútuo.

### Solução de problemas de avisos de certificado

Ao criar um nome de domínio personalizado com TLS mútuo, o API Gateway exibirá avisos se os certificados no armazenamento de confiança não forem válidos. Isso também pode ocorrer ao atualizar um nome de domínio personalizado para usar um novo armazenamento de confiança. Os

avisos indicam o problema com o certificado e o assunto que produziu o aviso. O TLS mútuo ainda está habilitado para sua API, mas alguns clientes podem não conseguir acessar a API.

Para identificar o certificado que produziu o aviso, é necessário decodificar os certificados em seu armazenamento de confiança. É possível usar ferramentas como `openssl` para decodificar os certificados e identificar os assuntos.

O comando a seguir exibe o conteúdo de um certificado, incluindo o assunto.

```
openssl x509 -in certificate.crt -text -noout
```

Atualize ou remova os certificados que produziram avisos e, depois, faça upload de um novo armazenamento de confiança para o Amazon S3. Após o upload do novo armazenamento de confiança, atualize o nome de domínio personalizado para usar o novo armazenamento de confiança.

### Solução de problemas de conflitos de nomes de domínios

O erro "The certificate subject <certSubject> conflicts with an existing certificate from a different issuer." significa que mais de uma autoridade de certificação emitiu um certificado para esse domínio. Para cada entidade no certificado, pode haver somente um emissor no API Gateway para domínios do TLS mútuos. Você precisará obter todos os seus certificados para esse assunto por meio de um único emissor. Se o problema for com um certificado sobre o qual você não tem controle, e você é capaz de provar que é o dono do nome de domínio, [entre em contato com Suporte](#) para abrir um ticket.

### Solução de problemas de mensagens de status de nomes de domínio

PENDING\_CERTIFICATE\_REIMPORT: isso significa que você reimportou um certificado para o ACM e ele falhou na validação porque o novo certificado tem um SAN (nome alternativo de entidade) que não é coberto pelo `ownershipVerificationCertificate` ou o assunto ou os SANs no certificado não cobrem o nome de domínio. Algo pode estar configurado incorretamente ou um certificado inválido foi importado. Você precisa reimportar um certificado válido para o ACM. Para obter mais informações sobre a validação, consulte [Validação da propriedade de domínios](#).

PENDING\_OWNERSHIP\_VERIFICATION: isso significa que seu certificado verificado anteriormente expirou e o ACM não conseguiu renová-lo automaticamente. Você precisará renovar o certificado ou solicitar um novo certificado. Mais informações sobre renovação de certificados podem ser encontradas no guia [Solução de problemas de renovação de certificados gerenciados do ACM](#).

## Solucionar problemas de certificado incorreto devolvido

Ao migrar um certificado dedicado de um nome de domínio totalmente qualificado (FQDN) para um nome de domínio de cliente curinga, o API Gateway pode retornar o certificado para o FQDN em vez do nome de domínio curinga.

O comando a seguir mostra qual certificado está sendo retornado pelo API Gateway:

```
openssl s_client -connect hostname:port
```

Se o certificado resultante for para o FQDN, [entre em contato com o Suporte](#) para abrir um tíquete.

## Gerar e configurar um certificado SSL para autenticação de backend no API Gateway

É possível usar o API Gateway para gerar um certificado SSL e, depois, usar sua chave pública no backend para verificar se as solicitações HTTP para seu sistema backend são provenientes do API Gateway. Isso permite que o backend HTTP controle e aceite apenas solicitações provenientes do Amazon API Gateway, mesmo que o backend esteja acessível ao público.

### Note

Alguns servidores de backend podem não ser compatíveis com a autenticação de cliente SSL como o API Gateway e retornar um erro de certificado SSL. Para obter uma lista de servidores de backend incompatíveis, consulte [the section called “Observações importantes”](#).

Os certificados SSL gerados pelo API Gateway são autoassinados, e somente a chave pública de um certificado é visível no console do API Gateway ou por meio das APIs.

### Tópicos

- [Gerar um certificado de cliente usando o console do API Gateway](#)
- [Configurar uma API para usar certificados SSL](#)
- [Testar chamada para verificar a configuração de certificado do cliente](#)
- [Configurar um servidor HTTPS de backend para verificar o certificado do cliente](#)
- [Girar um certificado de cliente prestes a expirar](#)
- [Autoridades de certificado compatíveis com o API Gateway para integrações HTTP e de proxy HTTP no API Gateway.](#)

## Gerar um certificado de cliente usando o console do API Gateway

1. Abra o console do API Gateway em <https://console.aws.amazon.com/apigateway/>.
2. Escolha uma API REST ou uma API de WebSocket.
3. No painel de navegação principal, selecione Certificados do cliente.
4. No painel Certificados do cliente, escolha Gerar certificado.
5. (Opcional) Em Description (Descrição), insira uma descrição.
6. Selecione Gerar certificado para gerar o certificado. O API Gateway gera um novo certificado e retorna o novo GUID de certificado, juntamente com a chave pública codificada em PEM.

Agora, você está pronto para configurar uma API para usar o certificado.

## Configurar uma API para usar certificados SSL

Essas instruções pressupõem que você já tenha concluído o [Gerar um certificado de cliente usando o console do API Gateway](#).

1. No console do API Gateway, crie ou abra uma API REST ou uma API de WebSocket para a qual você deseja usar o certificado de cliente. Certifique-se de que a API tenha sido implantada em um estágio.
2. No painel de navegação principal, selecione Estágios.
3. Na seção Detalhes do estágio, selecione Editar.
4. Em Certificado do cliente, selecione um certificado.
5. Escolha Salvar alterações.

Depois que um certificado for selecionado para a API e salvo, o API Gateway usará esse certificado para todas as chamadas para integrações HTTP em sua API.

## Testar chamada para verificar a configuração de certificado do cliente

1. Escolha um método de API REST. Selecione a guia Testar. Talvez seja necessário escolher o botão de seta para a direita para mostrar a guia Testar.
2. Em Certificado do cliente, selecione um certificado.
3. Escolha Testar.

O API Gateway apresenta o certificado SSL escolhido para que o backend HTTP autentique a API.

## Configurar um servidor HTTPS de backend para verificar o certificado do cliente

Estas instruções pressupõem que você já tenha concluído o [Gerar um certificado de cliente usando o console do API Gateway](#) e feito download de uma cópia do certificado do cliente. Você pode baixar um certificado de cliente chamando [clientcertificate:by-id](#) da API REST do API Gateway ou [get-client-certificate](#) da AWS CLI.

Antes de configurar um servidor HTTPS de backend para verificar o certificado SSL do cliente do API Gateway, é necessário ter obtido a chave privada codificada por PEM e um certificado de servidor fornecido por uma autoridade de certificação confiável.

Se o nome de domínio do servidor for `myserver.mydomain.com`, o valor CNAME do certificado do servidor deverá ser `myserver.mydomain.com` ou `*.mydomain.com`.

Entre as autoridades de certificação aceitas estão [Let's Encrypt](#) ou um dos [the section called "Autoridades de certificado com suporte para integração HTTP e de proxy HTTP"](#).

Como exemplo, suponha que o arquivo de certificado do cliente é `apig-cert.pem` e que a chave privada do servidor e os arquivos de certificado são `server-key.pem` e `server-cert.pem`, respectivamente. Para um servidor Node.js no backend, você pode configurar o servidor de forma semelhante à seguinte:

```
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 um aplicativo node-[express](#), você pode usar os módulos [client-certificate-auth](#) para autenticar solicitações de clientes com certificados codificados em PEM.

Para outros servidores HTTPS, consulte a documentação do servidor.

## Girar um certificado de cliente prestes a expirar

O certificado do cliente gerado pelo API Gateway é válido por 365 dias. Você deve girar o certificado antes que um certificado de cliente em um estágio de API expire, para evitar o tempo de inatividade da API.

Alternar um certificado de cliente prestes a expirar usando o Console de gerenciamento da AWS

O procedimento a seguir mostra como alternar um certificado de cliente no console para uma API implantada anteriormente.

1. No painel de navegação principal, selecione Certificados do cliente.
2. No painel Certificados do cliente, selecione Gerar certificado.
3. Abra a API para a qual você deseja usar o certificado do cliente.
4. Escolha Stages (Estágios) na API selecionada e selecione um estágio.
5. Na seção Detalhes do estágio, selecione Editar.
6. Em Certificado do cliente, selecione o novo certificado.
7. Para salvar as configurações, selecione Salvar alterações.

Alternar um certificado de cliente prestes a expirar usando o AWS CLI

Você pode verificar a data de validade do certificado chamando o [clientCertificate:by-id](#) da API REST do API Gateway ou o comando da AWS CLI [get-client-certificate](#) e inspecionando a propriedade [expirationDate](#) retornada.

Para alternar um certificado de cliente, faça o seguinte:

1. Para gerar um certificado de cliente, chame [clientcertificate:generate](#) da API REST do API Gateway ou o comando da AWS CLI de [generate-client-certificate](#). Neste tutorial, pressupomos que o novo ID de certificado do cliente é `ndiqef`.
2. Atualize o servidor de backend para incluir o novo certificado do cliente. Não remova o certificado de cliente existente ainda.

Alguns servidores podem exigir uma reinicialização para concluir a atualização. Consulte a documentação do servidor para ver se você deve reiniciá-lo durante a atualização.

3. Para atualizar o estágio da API para usar o novo certificado do cliente, chame [stage: update](#) da API REST do API Gateway, com o novo ID de certificado do 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"
 }
]
}
```

Também é possível usar o comando [update-stage](#).

Se você estiver utilizando uma API de WebSocket, use o comando `apigatewayv2` [update-stage](#).

4. Atualize o servidor de backend para remover o certificado antigo.
5. Para excluir o certificado antigo do API Gateway, chame [clientcertificate:delete](#) da API REST do API Gateway, especificando o `clientCertificateId` (a1b2c3) do certificado antigo:

```
DELETE /clientcertificates/a1b2c3
```

Também é possível chamar o comando [delete-client-certificate](#):

```
aws apigateway delete-client-certificate --client-certificate-id a1b2c3
```

Autoridades de certificado compatíveis com o API Gateway para integrações HTTP e de proxy HTTP no API Gateway.

A lista a seguir mostra as autoridades de certificação compatíveis com o API Gateway para integrações HTTP, de proxy HTTP e 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: atotrustedroot2011
SHA1: 2B:B1:F5:3E:55:0C:1D:C5:F1:D4:E6:B7:6A:46:4B:55:06:02:AC:21
SHA256:
F3:56:BE:A2:44:B7:A9:1E:B3:5D:53:CA:9A:D7:86:4A:CE:01:8E:2D:35:D5:F8:F9:6D:DF:68:A6:F4:1A:A4:7
Alias name: autoridaddecertificacionfirmaprofessionalcifa62634068
SHA1: AE:C5:FB:3F:C8:E1:BF:C4:E5:4F:03:07:5A:9A:E8:00:B7:F7:B6:FA
SHA256:
04:04:80:28:BF:1F:28:64:D4:8F:9A:D4:D8:32:94:36:6A:82:88:56:55:3F:3B:14:30:3F:90:14:7F:5D:40:E
Alias name: baltimorecodesigningca
SHA1: 30:46:D8:C8:88:FF:69:30:C3:4A:FC:CD:49:27:08:7C:60:56:7B:0D
SHA256:
A9:15:45:DB:D2:E1:9C:4C:CD:F9:09:AA:71:90:0D:18:C7:35:1C:89:B3:15:F0:F1:3D:05:C1:3A:8F:FB:46:8
Alias name: baltimorecybertrustca
SHA1: D4:DE:20:D0:5E:66:FC:53:FE:1A:50:88:2C:78:DB:28:52:CA:E4:74
SHA256:
16:AF:57:A9:F6:76:B0:AB:12:60:95:AA:5E:BA:DE:F2:2A:B3:11:19:D6:44:AC:95:CD:4B:93:DB:F3:F2:6A:E
Alias name: baltimorecybertrustroot
SHA1: D4:DE:20:D0:5E:66:FC:53:FE:1A:50:88:2C:78:DB:28:52:CA:E4:74
SHA256:
16:AF:57:A9:F6:76:B0:AB:12:60:95:AA:5E:BA:DE:F2:2A:B3:11:19:D6:44:AC:95:CD:4B:93:DB:F3:F2:6A:E
Alias name: buypassclass2ca
SHA1: 49:0A:75:74:DE:87:0A:47:FE:58:EE:F6:C7:6B:EB:C6:0B:12:40:99
SHA256:
9A:11:40:25:19:7C:5B:B9:5D:94:E6:3D:55:CD:43:79:08:47:B6:46:B2:3C:DF:11:AD:A4:A0:0E:FF:15:FB:4
Alias name: buypassclass2rootca
SHA1: 49:0A:75:74:DE:87:0A:47:FE:58:EE:F6:C7:6B:EB:C6:0B:12:40:99
SHA256:
9A:11:40:25:19:7C:5B:B9:5D:94:E6:3D:55:CD:43:79:08:47:B6:46:B2:3C:DF:11:AD:A4:A0:0E:FF:15:FB:4
Alias name: buypassclass3ca
SHA1: DA:FA:F7:FA:66:84:EC:06:8F:14:50:BD:C7:C2:81:A5:BC:A9:64:57
SHA256:
ED:F7:EB:BC:A2:7A:2A:38:4D:38:7B:7D:40:10:C6:66:E2:ED:B4:84:3E:4C:29:B4:AE:1D:5B:93:32:E6:B2:4
Alias name: buypassclass3rootca
SHA1: DA:FA:F7:FA:66:84:EC:06:8F:14:50:BD:C7:C2:81:A5:BC:A9:64:57
SHA256:
ED:F7:EB:BC:A2:7A:2A:38:4D:38:7B:7D:40:10:C6:66:E2:ED:B4:84:3E:4C:29:B4:AE:1D:5B:93:32:E6:B2:4
Alias name: cadisigrootr2
SHA1: B5:61:EB:EA:A4:DE:E4:25:4B:69:1A:98:A5:57:47:C2:34:C7:D9:71
```

```
SHA256:
E2:3D:4A:03:6D:7B:70:E9:F5:95:B1:42:20:79:D2:B9:1E:DF:BB:1F:B6:51:A0:63:3E:AA:8A:9D:C5:F8:07:0
Alias name: camerfirmachambersca
SHA1: 78:6A:74:AC:76:AB:14:7F:9C:6A:30:50:BA:9E:A8:7E:FE:9A:CE:3C
SHA256:
06:3E:4A:FA:C4:91:DF:D3:32:F3:08:9B:85:42:E9:46:17:D8:93:D7:FE:94:4E:10:A7:93:7E:E2:9D:96:93:C
Alias name: camerfirmachamberscommerceca
SHA1: 6E:3A:55:A4:19:0C:19:5C:93:84:3C:C0:DB:72:2E:31:30:61:F0:B1
SHA256:
0C:25:8A:12:A5:67:4A:EF:25:F2:8B:A7:DC:FA:EC:EE:A3:48:E5:41:E6:F5:CC:4E:E6:3B:71:B3:61:60:6A:C
Alias name: camerfirmachamberssignca
SHA1: 4A:BD:EE:EC:95:0D:35:9C:89:AE:C7:52:A1:2C:5B:29:F6:D6:AA:0C
SHA256:
13:63:35:43:93:34:A7:69:80:16:A0:D3:24:DE:72:28:4E:07:9D:7B:52:20:BB:8F:BD:74:78:16:EE:BE:BA:C
Alias name: certigna
SHA1: B1:2E:13:63:45:86:A4:6F:1A:B2:60:68:37:58:2D:C4:AC:FD:94:97
SHA256:
E3:B6:A2:DB:2E:D7:CE:48:84:2F:7A:C5:32:41:C7:B7:1D:54:14:4B:FB:40:C1:1F:3F:1D:0B:42:F5:EE:A1:2
Alias name: certignarootca
SHA1: 2D:0D:52:14:FF:9E:AD:99:24:01:74:20:47:6E:6C:85:27:27:F5:43
SHA256:
D4:8D:3D:23:EE:DB:50:A4:59:E5:51:97:60:1C:27:77:4B:9D:7B:18:C9:4D:5A:05:95:11:A1:02:50:B9:31:6
Alias name: certplusclass2primaryca
SHA1: 74:20:74:41:72:9C:DD:92:EC:79:31:D8:23:10:8D:C2:81:92:E2:BB
SHA256:
0F:99:3C:8A:EF:97:BA:AF:56:87:14:0E:D5:9A:D1:82:1B:B4:AF:AC:F0:AA:9A:58:B5:D5:7A:33:8A:3A:FB:C
Alias name: certplusclass3ppprimaryca
SHA1: 21:6B:2A:29:E6:2A:00:CE:82:01:46:D8:24:41:41:B9:25:11:B2:79
SHA256:
CC:C8:94:89:37:1B:AD:11:1C:90:61:9B:EA:24:0A:2E:6D:AD:D9:9F:9F:6E:1D:4D:41:E5:8E:D6:DE:3D:02:8
Alias name: certsignrootca
SHA1: FA:B7:EE:36:97:26:62:FB:2D:B0:2A:F6:BF:03:FD:E8:7C:4B:2F:9B
SHA256:
EA:A9:62:C4:FA:4A:6B:AF:EB:E4:15:19:6D:35:1C:CD:88:8D:4F:53:F3:FA:8A:E6:D7:C4:66:A9:4E:60:42:B
Alias name: certsignrootcag2
SHA1: 26:F9:93:B4:ED:3D:28:27:B0:B9:4B:A7:E9:15:1D:A3:8D:92:E5:32
SHA256:
65:7C:FE:2F:A7:3F:AA:38:46:25:71:F3:32:A2:36:3A:46:FC:E7:02:09:51:71:07:02:CD:FB:B6:EE:DA:33:0
Alias name: certum2
SHA1: D3:DD:48:3E:2B:BF:4C:05:E8:AF:10:F5:FA:76:26:CF:D3:DC:30:92
SHA256:
B6:76:F2:ED:DA:E8:77:5C:D3:6C:B0:F6:3C:D1:D4:60:39:61:F4:9E:62:65:BA:01:3A:2F:03:07:B6:D0:B8:0
Alias name: certumca
SHA1: 62:52:DC:40:F7:11:43:A2:2F:DE:9E:F7:34:8E:06:42:51:B1:81:18
```

```
SHA256:
D8:E0:FE:BC:1D:B2:E3:8D:00:94:0F:37:D2:7D:41:34:4D:99:3E:73:4B:99:D5:65:6D:97:78:D4:D8:14:36:2
Alias name: certumtrustednetworkca
SHA1: 07:E0:32:E0:20:B7:2C:3F:19:2F:06:28:A2:59:3A:19:A7:0F:06:9E
SHA256:
5C:58:46:8D:55:F5:8E:49:7E:74:39:82:D2:B5:00:10:B6:D1:65:37:4A:CF:83:A7:D4:A3:2D:B7:68:C4:40:8
Alias name: certumtrustednetworkca2
SHA1: D3:DD:48:3E:2B:BF:4C:05:E8:AF:10:F5:FA:76:26:CF:D3:DC:30:92
SHA256:
B6:76:F2:ED:DA:E8:77:5C:D3:6C:B0:F6:3C:D1:D4:60:39:61:F4:9E:62:65:BA:01:3A:2F:03:07:B6:D0:B8:0
Alias name: cfcaevroot
SHA1: E2:B8:29:4B:55:84:AB:6B:58:C2:90:46:6C:AC:3F:B8:39:8F:84:83
SHA256:
5C:C3:D7:8E:4E:1D:5E:45:54:7A:04:E6:87:3E:64:F9:0C:F9:53:6D:1C:CC:2E:F8:00:F3:55:C4:C5:FD:70:F
Alias name: chambersofcommerceroott2008
SHA1: 78:6A:74:AC:76:AB:14:7F:9C:6A:30:50:BA:9E:A8:7E:FE:9A:CE:3C
SHA256:
06:3E:4A:FA:C4:91:DF:D3:32:F3:08:9B:85:42:E9:46:17:D8:93:D7:FE:94:4E:10:A7:93:7E:E2:9D:96:93:C
Alias name: chungwaepkirootca
SHA1: 67:65:0D:F1:7E:8E:7E:5B:82:40:A4:F4:56:4B:CF:E2:3D:69:C6:F0
SHA256:
C0:A6:F4:DC:63:A2:4B:FD:CF:54:EF:2A:6A:08:2A:0A:72:DE:35:80:3E:2F:F5:FF:52:7A:E5:D8:72:06:DF:D
Alias name: cia-crt-g3-01-ca
SHA1: 2B:EE:2C:BA:A3:1D:B5:FE:60:40:41:95:08:ED:46:82:39:4D:ED:E2
SHA256:
20:48:AD:4C:EC:90:7F:FA:4A:15:D4:CE:45:E3:C8:E4:2C:EA:78:33:DC:C7:D3:40:48:FC:60:47:27:42:99:E
Alias name: cia-crt-g3-02-ca
SHA1: 96:4A:BB:A7:BD:DA:FC:97:34:C0:0A:2D:F0:05:98:F7:E6:C6:6F:09
SHA256:
93:F1:72:FB:BA:43:31:5C:06:EE:0F:9F:04:89:B8:F6:88:BC:75:15:3C:BE:B4:80:AC:A7:14:3A:F6:FC:4A:C
Alias name: comodo-ca
SHA1: AF:E5:D2:44:A8:D1:19:42:30:FF:47:9F:E2:F8:97:BB:CD:7A:8C:B4
SHA256:
52:F0:E1:C4:E5:8E:C6:29:29:1B:60:31:7F:07:46:71:B8:5D:7E:A8:0D:5B:07:27:34:63:53:4B:32:B4:02:3
Alias name: comodoaaaca
SHA1: D1:EB:23:A4:6D:17:D6:8F:D9:25:64:C2:F1:F1:60:17:64:D8:E3:49
SHA256:
D7:A7:A0:FB:5D:7E:27:31:D7:71:E9:48:4E:BC:DE:F7:1D:5F:0C:3E:0A:29:48:78:2B:C8:3E:E0:EA:69:9E:F
Alias name: comodoaaaservicesroot
SHA1: D1:EB:23:A4:6D:17:D6:8F:D9:25:64:C2:F1:F1:60:17:64:D8:E3:49
SHA256:
D7:A7:A0:FB:5D:7E:27:31:D7:71:E9:48:4E:BC:DE:F7:1D:5F:0C:3E:0A:29:48:78:2B:C8:3E:E0:EA:69:9E:F
Alias name: comodocertificationauthority
SHA1: 66:31:BF:9E:F7:4F:9E:B6:C9:D5:A6:0C:BA:6A:BE:D1:F7:BD:EF:7B
```

```
SHA256:
0C:2C:D6:3D:F7:80:6F:A3:99:ED:E8:09:11:6B:57:5B:F8:79:89:F0:65:18:F9:80:8C:86:05:03:17:8B:AF:6
Alias name: comodoecccertificationauthority
SHA1: 9F:74:4E:9F:2B:4D:BA:EC:0F:31:2C:50:B6:56:3B:8E:2D:93:C3:11
SHA256:
17:93:92:7A:06:14:54:97:89:AD:CE:2F:8F:34:F7:F0:B6:6D:0F:3A:E3:A3:B8:4D:21:EC:15:DB:BA:4F:AD:C
Alias name: comodorsacertificationauthority
SHA1: AF:E5:D2:44:A8:D1:19:42:30:FF:47:9F:E2:F8:97:BB:CD:7A:8C:B4
SHA256:
52:F0:E1:C4:E5:8E:C6:29:29:1B:60:31:7F:07:46:71:B8:5D:7E:A8:0D:5B:07:27:34:63:53:4B:32:B4:02:3
Alias name: cybertrustglobalroot
SHA1: 5F:43:E5:B1:BF:F8:78:8C:AC:1C:C7:CA:4A:9A:C6:22:2B:CC:34:C6
SHA256:
96:0A:DF:00:63:E9:63:56:75:0C:29:65:DD:0A:08:67:DA:0B:9C:BD:6E:77:71:4A:EA:FB:23:49:AB:39:3D:A
Alias name: deprecateditsecca
SHA1: 12:12:0B:03:0E:15:14:54:F4:DD:B3:F5:DE:13:6E:83:5A:29:72:9D
SHA256:
9A:59:DA:86:24:1A:FD:BA:A3:39:FA:9C:FD:21:6A:0B:06:69:4D:E3:7E:37:52:6B:BE:63:C8:BC:83:74:2E:C
Alias name: deutschetelekomrootca2
SHA1: 85:A4:08:C0:9C:19:3E:5D:51:58:7D:CD:D6:13:30:FD:8C:DE:37:BF
SHA256:
B6:19:1A:50:D0:C3:97:7F:7D:A9:9B:CD:AA:C8:6A:22:7D:AE:B9:67:9E:C7:0B:A3:B0:C9:D9:22:71:C1:70:D
Alias name: digicertassuredidrootca
SHA1: 05:63:B8:63:0D:62:D7:5A:BB:C8:AB:1E:4B:DF:B5:A8:99:B2:4D:43
SHA256:
3E:90:99:B5:01:5E:8F:48:6C:00:BC:EA:9D:11:1E:E7:21:FA:BA:35:5A:89:BC:F1:DF:69:56:1E:3D:C6:32:5
Alias name: digicertassuredidrootg2
SHA1: A1:4B:48:D9:43:EE:0A:0E:40:90:4F:3C:E0:A4:C0:91:93:51:5D:3F
SHA256:
7D:05:EB:B6:82:33:9F:8C:94:51:EE:09:4E:EB:FE:FA:79:53:A1:14:ED:B2:F4:49:49:45:2F:AB:7D:2F:C1:8
Alias name: digicertassuredidrootg3
SHA1: F5:17:A2:4F:9A:48:C6:C9:F8:A2:00:26:9F:DC:0F:48:2C:AB:30:89
SHA256:
7E:37:CB:8B:4C:47:09:0C:AB:36:55:1B:A6:F4:5D:B8:40:68:0F:BA:16:6A:95:2D:B1:00:71:7F:43:05:3F:C
Alias name: digicertglobalrootca
SHA1: A8:98:5D:3A:65:E5:E5:C4:B2:D7:D6:6D:40:C6:DD:2F:B1:9C:54:36
SHA256:
43:48:A0:E9:44:4C:78:CB:26:5E:05:8D:5E:89:44:B4:D8:4F:96:62:BD:26:DB:25:7F:89:34:A4:43:C7:01:6
Alias name: digicertglobalrootg2
SHA1: DF:3C:24:F9:BF:D6:66:76:1B:26:80:73:FE:06:D1:CC:8D:4F:82:A4
SHA256:
CB:3C:CB:B7:60:31:E5:E0:13:8F:8D:D3:9A:23:F9:DE:47:FF:C3:5E:43:C1:14:4C:EA:27:D4:6A:5A:B1:CB:5
Alias name: digicertglobalrootg3
SHA1: 7E:04:DE:89:6A:3E:66:6D:00:E6:87:D3:3F:FA:D9:3B:E8:3D:34:9E
```

```

SHA256:
31:AD:66:48:F8:10:41:38:C7:38:F3:9E:A4:32:01:33:39:3E:3A:18:CC:02:29:6E:F9:7C:2A:C9:EF:67:31:D
Alias name: digicerthighassuranceevrootca
SHA1: 5F:B7:EE:06:33:E2:59:DB:AD:0C:4C:9A:E6:D3:8F:1A:61:C7:DC:25
SHA256:
74:31:E5:F4:C3:C1:CE:46:90:77:4F:0B:61:E0:54:40:88:3B:A9:A0:1E:D0:0B:A6:AB:D7:80:6E:D3:B1:18:C
Alias name: digicerttrustedrootg4
SHA1: DD:FB:16:CD:49:31:C9:73:A2:03:7D:3F:C8:3A:4D:7D:77:5D:05:E4
SHA256:
55:2F:7B:DC:F1:A7:AF:9E:6C:E6:72:01:7F:4F:12:AB:F7:72:40:C7:8E:76:1A:C2:03:D1:D9:D2:0A:C8:99:8
Alias name: 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: mozillacert50.pem
SHA1: B8:01:86:D1:EB:9C:86:A5:41:04:CF:30:54:F3:4C:52:B7:E5:58:C6
SHA256:
CE:CD:DC:90:50:99:D8:DA:DF:C5:B1:D2:09:B7:37:CB:E2:C1:8C:FB:2C:10:C0:FF:0B:CF:0D:32:86:FC:1A:A
Alias name: mozillacert51.pem
SHA1: 8C:96:BA:EB:DD:2B:07:07:48:EE:30:32:66:A0:F3:98:6E:7C:AE:58
SHA256:
35:AE:5B:DD:D8:F7:AE:63:5C:FF:BA:56:82:A8:F0:0B:95:F4:84:62:C7:10:8E:E9:A0:E5:29:2B:07:4A:AF:B
Alias name: mozillacert52.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
```

```
SHA256:
E2:83:93:77:3D:A8:45:A6:79:F2:08:0C:C7:FB:44:A3:B7:A1:C3:79:2C:B7:EB:77:29:FD:CB:6A:8D:99:AE:A
Alias name: mozillacert53.pem
SHA1: 7F:8A:B0:CF:D0:51:87:6A:66:F3:36:0F:47:C8:8D:8C:D3:35:FC:74
SHA256:
2D:47:43:7D:E1:79:51:21:5A:12:F3:C5:8E:51:C7:29:A5:80:26:EF:1F:CC:0A:5F:B3:D9:DC:01:2F:60:0D:1
Alias name: mozillacert54.pem
SHA1: 03:9E:ED:B8:0B:E7:A0:3C:69:53:89:3B:20:D2:D9:32:3A:4C:2A:FD
SHA256:
B4:78:B8:12:25:0D:F8:78:63:5C:2A:A7:EC:7D:15:5E:AA:62:5E:E8:29:16:E2:CD:29:43:61:88:6C:D1:FB:D
Alias name: mozillacert55.pem
SHA1: AA:DB:BC:22:23:8F:C4:01:A1:27:BB:38:DD:F4:1D:DB:08:9E:F0:12
SHA256:
A4:31:0D:50:AF:18:A6:44:71:90:37:2A:86:AF:AF:8B:95:1F:FB:43:1D:83:7F:1E:56:88:B4:59:71:ED:15:5
Alias name: mozillacert56.pem
SHA1: F1:8B:53:8D:1B:E9:03:B6:A6:F0:56:43:5B:17:15:89:CA:F3:6B:F2
SHA256:
4B:03:F4:58:07:AD:70:F2:1B:FC:2C:AE:71:C9:FD:E4:60:4C:06:4C:F5:FF:B6:86:BA:E5:DB:AA:D7:FD:D3:4
Alias name: mozillacert57.pem
SHA1: D6:DA:A8:20:8D:09:D2:15:4D:24:B5:2F:CB:34:6E:B2:58:B2:8A:58
SHA256:
F9:E6:7D:33:6C:51:00:2A:C0:54:C6:32:02:2D:66:DD:A2:E7:E3:FF:F1:0A:D0:61:ED:31:D8:BB:B4:10:CF:B
Alias name: mozillacert58.pem
SHA1: 8D:17:84:D5:37:F3:03:7D:EC:70:FE:57:8B:51:9A:99:E6:10:D7:B0
SHA256:
5E:DB:7A:C4:3B:82:A0:6A:87:61:E8:D7:BE:49:79:EB:F2:61:1F:7D:D7:9B:F9:1C:1C:6B:56:6A:21:9E:D7:6
Alias name: mozillacert59.pem
SHA1: 36:79:CA:35:66:87:72:30:4D:30:A5:FB:87:3B:0F:A7:7B:B7:0D:54
SHA256:
23:99:56:11:27:A5:71:25:DE:8C:EF:EA:61:0D:DF:2F:A0:78:B5:C8:06:7F:4E:82:82:90:BF:B8:60:E8:4B:3
Alias name: mozillacert60.pem
SHA1: 27:96:BA:E6:3F:18:01:E2:77:26:1B:A0:D7:77:70:02:8F:20:EE:E4
SHA256:
C3:84:6B:F2:4B:9E:93:CA:64:27:4C:0E:C6:7C:1E:CC:5E:02:4F:FC:AC:D2:D7:40:19:35:0E:81:FE:54:6A:E
Alias name: mozillacert61.pem
SHA1: 3B:C4:9F:48:F8:F3:73:A0:9C:1E:BD:F8:5B:B1:C3:65:C7:D8:11:B3
SHA256:
BF:0F:EE:FB:9E:3A:58:1A:D5:F9:E9:DB:75:89:98:57:43:D2:61:08:5C:4D:31:4F:6F:5D:72:59:AA:42:16:1
Alias name: mozillacert62.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: mozillacert63.pem
SHA1: A1:DB:63:93:91:6F:17:E4:18:55:09:40:04:15:C7:02:40:B0:AE:6B
```

```
SHA256:
A4:B6:B3:99:6F:C2:F3:06:B3:FD:86:81:BD:63:41:3D:8C:50:09:CC:4F:A3:29:C2:CC:F0:E2:FA:1B:14:03:0
Alias name: mozillacert63.pem
SHA1: 89:DF:74:FE:5C:F4:0F:4A:80:F9:E3:37:7D:54:DA:91:E1:01:31:8E
SHA256:
3C:5F:81:FE:A5:FA:B8:2C:64:BF:A2:EA:EC:AF:CD:E8:E0:77:FC:86:20:A7:CA:E5:37:16:3D:F3:6E:DB:F3:7
Alias name: mozillacert64.pem
SHA1: 62:7F:8D:78:27:65:63:99:D2:7D:7F:90:44:C9:FE:B3:F3:3E:FA:9A
SHA256:
AB:70:36:36:5C:71:54:AA:29:C2:C2:9F:5D:41:91:16:3B:16:2A:22:25:01:13:57:D5:6D:07:FF:A7:BC:1F:7
Alias name: mozillacert65.pem
SHA1: 69:BD:8C:F4:9C:D3:00:FB:59:2E:17:93:CA:55:6A:F3:EC:AA:35:FB
SHA256:
BC:23:F9:8A:31:3C:B9:2D:E3:BB:FC:3A:5A:9F:44:61:AC:39:49:4C:4A:E1:5A:9E:9D:F1:31:E9:9B:73:01:9
Alias name: mozillacert66.pem
SHA1: DD:E1:D2:A9:01:80:2E:1D:87:5E:84:B3:80:7E:4B:B1:FD:99:41:34
SHA256:
E6:09:07:84:65:A4:19:78:0C:B6:AC:4C:1C:0B:FB:46:53:D9:D9:CC:6E:B3:94:6E:B7:F3:D6:99:97:BA:D5:9
Alias name: mozillacert67.pem
SHA1: D6:9B:56:11:48:F0:1C:77:C5:45:78:C1:09:26:DF:5B:85:69:76:AD
SHA256:
CB:B5:22:D7:B7:F1:27:AD:6A:01:13:86:5B:DF:1C:D4:10:2E:7D:07:59:AF:63:5A:7C:F4:72:0D:C9:63:C5:3
Alias name: mozillacert68.pem
SHA1: AE:C5:FB:3F:C8:E1:BF:C4:E5:4F:03:07:5A:9A:E8:00:B7:F7:B6:FA
SHA256:
04:04:80:28:BF:1F:28:64:D4:8F:9A:D4:D8:32:94:36:6A:82:88:56:55:3F:3B:14:30:3F:90:14:7F:5D:40:E
Alias name: mozillacert69.pem
SHA1: 2F:78:3D:25:52:18:A7:4A:65:39:71:B5:2C:A2:9C:45:15:6F:E9:19
SHA256:
25:30:CC:8E:98:32:15:02:BA:D9:6F:9B:1F:BA:1B:09:9E:2D:29:9E:0F:45:48:BB:91:4F:36:3B:C0:D4:53:1
Alias name: mozillacert70.pem
SHA1: AD:7E:1C:28:B0:64:EF:8F:60:03:40:20:14:C3:D0:E3:37:0E:B5:8A
SHA256:
14:65:FA:20:53:97:B8:76:FA:A6:F0:A9:95:8E:55:90:E4:0F:CC:7F:AA:4F:B7:C2:C8:67:75:21:FB:5F:B6:5
Alias name: mozillacert71.pem
SHA1: 4A:BD:EE:EC:95:0D:35:9C:89:AE:C7:52:A1:2C:5B:29:F6:D6:AA:0C
SHA256:
13:63:35:43:93:34:A7:69:80:16:A0:D3:24:DE:72:28:4E:07:9D:7B:52:20:BB:8F:BD:74:78:16:EE:BE:BA:C
Alias name: mozillacert72.pem
SHA1: 47:BE:AB:C9:22:EA:E8:0E:78:78:34:62:A7:9F:45:C2:54:FD:E6:8B
```



```
SHA256:
45:14:0B:32:47:EB:9C:C8:C5:B4:F0:D7:B5:30:91:F7:32:92:08:9E:6E:5A:63:E2:74:9D:D3:AC:A9:19:8E:D
Alias name: mozillacert73.pem
SHA1: B5:1C:06:7C:EE:2B:0C:3D:F8:55:AB:2D:92:F4:FE:39:D4:E7:0F:0E
SHA256:
2C:E1:CB:0B:F9:D2:F9:E1:02:99:3F:BE:21:51:52:C3:B2:DD:0C:AB:DE:1C:68:E5:31:9B:83:91:54:DB:B7:F
Alias name: mozillacert74.pem
SHA1: 92:5A:8F:8D:2C:6D:04:E0:66:5F:59:6A:FF:22:D8:63:E8:25:6F:3F
SHA256:
56:8D:69:05:A2:C8:87:08:A4:B3:02:51:90:ED:CF:ED:B1:97:4A:60:6A:13:C6:E5:29:0F:CB:2A:E6:3E:DA:B
Alias name: mozillacert75.pem
SHA1: D2:32:09:AD:23:D3:14:23:21:74:E4:0D:7F:9D:62:13:97:86:63:3A
SHA256:
08:29:7A:40:47:DB:A2:36:80:C7:31:DB:6E:31:76:53:CA:78:48:E1:BE:BD:3A:0B:01:79:A7:07:F9:2C:F1:7
Alias name: mozillacert76.pem
SHA1: F9:B5:B6:32:45:5F:9C:BE:EC:57:5F:80:DC:E9:6E:2C:C7:B2:78:B7
SHA256:
03:76:AB:1D:54:C5:F9:80:3C:E4:B2:E2:01:A0:EE:7E:EF:7B:57:B6:36:E8:A9:3C:9B:8D:48:60:C9:6F:5F:A
Alias name: mozillacert77.pem
SHA1: 13:2D:0D:45:53:4B:69:97:CD:B2:D5:C3:39:E2:55:76:60:9B:5C:C6
SHA256:
EB:04:CF:5E:B1:F3:9A:FA:76:2F:2B:B1:20:F2:96:CB:A5:20:C1:B9:7D:B1:58:95:65:B8:1C:B9:A1:7B:72:4
Alias name: mozillacert78.pem
SHA1: 29:36:21:02:8B:20:ED:02:F5:66:C5:32:D1:D6:ED:90:9F:45:00:2F
SHA256:
0A:81:EC:5A:92:97:77:F1:45:90:4A:F3:8D:5D:50:9F:66:B5:E2:C5:8F:CD:B5:31:05:8B:0E:17:F3:F0:B4:1
Alias name: mozillacert79.pem
SHA1: D8:A6:33:2C:E0:03:6F:B1:85:F6:63:4F:7D:6A:06:65:26:32:28:27
SHA256:
70:A7:3F:7F:37:6B:60:07:42:48:90:45:34:B1:14:82:D5:BF:0E:69:8E:CC:49:8D:F5:25:77:EB:F2:E9:3B:9
Alias name: 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: soneraiclass1ca
SHA1: 07:47:22:01:99:CE:74:B9:7C:B0:3D:79:B2:64:A2:C8:55:E9:33:FF
SHA256:
CD:80:82:84:CF:74:6F:F2:FD:6E:B5:8A:A1:D5:9C:4A:D4:B3:CA:56:FD:C6:27:4A:89:26:A7:83:5F:32:31:3
Alias name: soneraiclass2ca
SHA1: 37:F7:6D:E6:07:7C:90:C5:B1:3E:93:1A:B7:41:10:B4:F2:E4:9A:27
SHA256:
79:08:B4:03:14:C1:38:10:0B:51:8D:07:35:80:7F:FB:FC:F8:51:8A:00:95:33:71:05:BA:38:6B:15:3D:D9:2
Alias name: soneraiclass2rootca
SHA1: 37:F7:6D:E6:07:7C:90:C5:B1:3E:93:1A:B7:41:10:B4:F2:E4:9A:27
SHA256:
79:08:B4:03:14:C1:38:10:0B:51:8D:07:35:80:7F:FB:FC:F8:51:8A:00:95:33:71:05:BA:38:6B:15:3D:D9:2
Alias name: sslcomevrootcertificationauthorityecc
SHA1: 4C:DD:51:A3:D1:F5:20:32:14:B0:C6:C5:32:23:03:91:C7:46:42:6D
SHA256:
22:A2:C1:F7:BD:ED:70:4C:C1:E7:01:B5:F4:08:C3:10:88:0F:E9:56:B5:DE:2A:4A:44:F9:9C:87:3A:25:A7:C
Alias name: 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: swissigngoldcag2
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: swissigngoldg2ca
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: swissignplatinumg2ca
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: trustcorecal
SHA1: 58:D1:DF:95:95:67:6B:63:C0:F0:5B:1C:17:4D:8B:84:0B:C8:78:BD
SHA256:
5A:88:5D:B1:9C:01:D9:12:C5:75:93:88:93:8C:AF:BB:DF:03:1A:B2:D4:8E:91:EE:15:58:9B:42:97:1D:03:9
Alias name: 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
```

## Usar o AWS WAF para proteger as APIs REST no API Gateway

AWS WAFO é um firewall de aplicações que ajuda a proteger aplicações e APIs Web contra ataques. Isso permite configurar um conjunto de regras chamado de lista de controle de acesso à web (ACL da web) que permitem, bloqueiam ou contam solicitações da web com base em regras e condições de segurança da web personalizáveis que você define. Para obter mais informações, consulte [Como o AWS WAF funciona](#).

É possível usar o AWS WAF para proteger a API REST do API Gateway contra explorações comuns da Web, como injeção de SQL e ataques de cross-site scripting (XSS). Isso pode afetar a disponibilidade e a performance da API, comprometer a segurança ou consumir recursos excessivos. Por exemplo, você pode criar regras para permitir ou bloquear solicitações de intervalos de endereços IP especificados, solicitações de blocos CIDR, solicitações originárias de um país ou região específico, solicitações que contenham código SQL mal-intencionado ou solicitações que contenham script mal-intencionado.

Você também pode criar regras que correspondam a uma string especificada ou um padrão de expressão regular em cabeçalhos HTTP, método, URI, string de consulta e o corpo da solicitação (limitados aos primeiros 64 KB). Além disso, você pode criar regras para bloquear ataques de agentes de usuário específicos, bad bots e descarte de conteúdo. Por exemplo, podem ser utilizadas regras baseadas em taxa para especificar o número de solicitações da web que são permitidas por cada IP do cliente no final de um período de cinco minutos em atualização contínua.

### Important

AWS WAFO é a primeira linha de defesa contra explorações da web. Quando o AWS WAF está habilitado em uma API, as regras do AWS WAF são avaliadas antes de outros recursos de controle de acesso, como [políticas de recursos](#), [políticas do IAM](#), [autorizadores do Lambda](#) e [autorizadores do Amazon Cognito](#). Por exemplo, se o AWS WAF bloquear

o acesso de um bloco CIDR que uma política de recurso permite, o AWS WAF terá precedência e a política de recurso não será avaliada.

Para habilitar o AWS WAF para sua API, você precisa fazer o seguinte:

1. Use o console do AWS WAF, o AWS SDK ou a CLI para criar uma ACL da web que contenha a combinação desejada de regras gerenciadas do AWS WAF e suas próprias regras personalizadas. Para saber mais, consulte [Getting Started with AWS WAF](#) e [Web access control lists \(web ACLs\)](#).

 Important

O API Gateway exige uma ACL da web do AWS WAFV2 para uma aplicação regional ou uma ACL da web do AWS WAF Classic regional.

2. Associar uma ACL da web do AWS WAF a um estágio de API. É possível fazer isso usando o console do AWS WAF, o AWS SDK ou a CLI ou usando o console do API Gateway.

## Associar uma ACL da web do AWS WAF a um estágio da API do API Gateway usando o console do API Gateway

Para usar o console do API Gateway a fim de associar uma ACL da web do AWS WAF a um estágio da API do API Gateway existente, use as seguintes etapas:

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API existente ou crie uma nova.
3. No painel de navegação principal, selecione Estágios e escolha um estágio.
4. Na seção Detalhes do estágio, selecione Editar.
5. Em Firewall de aplicativo Web (AWS WAF), selecione sua ACL da web.

Se você estiver usando o AWS WAFV2, selecione uma ACL da web do AWS WAFV2 para uma aplicação regional. A ACL da web e quaisquer outros recursos do AWS WAFV2 que ela usa devem estar localizados na mesma região que sua API.

Se você estiver usando o AWS WAF Classic regional, selecione a ACL da web regional.

6. Escolha Salvar alterações.

## Associar uma ACL da web do AWS WAF a um estágio da API do API Gateway usando a AWS CLI

O comando [associate-web-acl](#) indicado abaixo associa uma ACL da web do AWS WAFV2 para uma aplicação regional a um estágio de API do 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
```

O comando [associate-web-acl](#) indicado abaixo associa uma ACL da web do AWS WAF Classic regional a um estágio de API do 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'
```

## Associar uma ACL da web do AWS WAF a um estágio da API usando a API REST do AWS WAF

Para usar a API REST do AWS WAFV2 a fim de associar uma ACL da web do AWS WAFV2 de uma aplicação regional a um estágio da API do API Gateway, use o comando [AssociateWebACL](#), como no seguinte exemplo:

```
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'
)
```

Para usar a API REST do AWS WAF a fim de associar uma ACL da web do AWS WAF Classic regional a um estágio da API do API Gateway, use o comando [AssociateWebACL](#), como no seguinte exemplo:

```
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'
)
```

## Controle de utilização das solicitações das APIs REST para ter um melhor throughput no API Gateway

Você pode configurar o controle de utilização e cotas para as suas APIs para ajudar a protegê-las da sobrecarga de numerosas solicitações. Os controles de utilização e as cotas são aplicados de acordo com o melhor esforço e devem ser considerados alvos, e não limites máximos garantidos de solicitações.

O API Gateway controla a utilização das solicitações para a sua API usando o algoritmo do bucket de token, em que um token equivale a uma solicitação. Especificamente, o API Gateway analisa a taxa e uma intermitência de envios de solicitações de todas as APIs na sua conta, por região. No algoritmo do bucket de token, uma intermitência pode permitir a saturação predefinida desses limites, mas há alguns casos em que outros fatores também podem exceder tais limites.

Quando os envios de solicitações excederem a taxa de solicitação de estado fixo e os limites de intermitência, o API Gateway iniciará o controle de utilização de solicitações. Neste momento, pode ser que os clientes recebam respostas de erro 429 Too Many Requests. Ao capturar essas exceções, o cliente poderá reenviar as solicitações com falha de uma forma que restrinja as taxas.

Como desenvolvedor de APIs, você pode definir os limites alvo para estágios ou métodos de API particulares para melhorar a performance geral em todas as APIs na sua conta. Como alternativa, você pode habilitar planos de uso para configurar controles de utilização aos envios de solicitações do cliente com base nas cotas e taxas de solicitação especificadas.

### Tópicos

- [Como as configurações de controle de utilização são aplicadas no API Gateway](#)
- [Controle de utilização no nível da conta por região](#)
- [Configuração de alvos de controle de utilização no nível da API e no nível do estágio em um plano de uso](#)
- [Configurar destinos de controle de utilização no nível do estágio](#)



- [Configuração de alvos de controle de utilização no nível do método em um plano de uso](#)

## Como as configurações de controle de utilização são aplicadas no API Gateway

Antes de definir as configurações de controle de utilização e cotas para a API, é útil entender os tipos de configurações relacionadas ao controle de utilização para a API e como o API Gateway as aplica.

O Amazon API Gateway fornece quatro tipos básicos de configurações relacionadas ao controle de utilização:

- Os limites de controle de utilização da AWS são aplicados em todas as contas e clientes de uma região. Essas configurações de limite existem para impedir que sua API e sua conta sejam sobrecarregadas com muitas solicitações. Tais limites são definidos pela AWS e não poderão ser alterados por um cliente.
- Os limites por conta são aplicados a todas as APIs em uma conta em uma região especificada. O limite de taxas no nível da conta pode ser aumentado por meio de uma solicitação; é possível obter limites mais altos com APIs com tempos limite mais curtos e cargas úteis menores. Para solicitar um aumento nos limites de controle de utilização no nível da conta por região, entre em contato com a [Central de Suporte da AWS](#). Para obter mais informações, consulte [Cotas e observações importantes](#). Observe que tais limites não podem ser superiores aos limites de controle de utilização da AWS.
- Os limites de controle de utilização por API e por estágio são aplicados no nível do método da API para um estágio. Você pode definir as mesmas configurações para todos os métodos ou configurações de controle de utilização distintas para cada método. Observe que tais limites não podem ser superiores aos limites de controle de utilização da AWS.
- Os limites de controle de utilização por cliente são aplicados aos clientes que usam chaves de API associadas ao seu plano de uso como identificador de cliente. Observe que tais limites não podem ser superiores aos limites por conta.

O API Gateway aplica as configurações relacionadas ao controle de utilização na seguinte ordem:

1. [Limites de controle de utilização por cliente ou por método](#) que você define para um estágio de API em um [plano de uso](#)
2. [Os limites de controle de utilização por método que você define para um estágio da API](#)
3. [Controle de utilização no nível da conta por região](#)
4. AWSControle de utilização regional da

## Controle de utilização no nível da conta por região

Por padrão, o API Gateway controla a utilização das solicitações de estado fixo por segundo (RPS) em todas as APIs de uma conta da AWS, por região. Ele também limita a intermitência (ou seja, o tamanho máximo do bucket) em todas as APIs de uma conta da AWS, por região. No API Gateway, o limite de intermitência representa o número máximo alvo de envios simultâneos de solicitações que ele fará antes de retornar respostas de erro 429 Too Many Requests. Para obter mais informações sobre cotas de controle de utilização, consulte [Cotas e observações importantes](#).

## Configuração de alvos de controle de utilização no nível da API e no nível do estágio em um plano de uso

Em um [plano de uso](#), é possível definir um destino de controle de utilização por método para todos os métodos em nível de API ou de estágio. É possível especificar uma taxa de controle de utilização, que é a taxa, em solicitações por segundo, em que os tokens são adicionados ao bucket do token. Você também pode especificar uma expansão do controle de utilização, que é a capacidade do bucket do token.

Você pode usar a AWS CLI, SDKs e o Console de gerenciamento da AWS para criar um plano de uso. Para saber mais sobre como criar um plano de uso, consulte [???](#).

## Configurar destinos de controle de utilização no nível do estágio

É possível usar a AWS CLI, os SDKs e o Console de gerenciamento da AWS para criar destinos de controle de utilização no nível de estágio.

Para saber mais sobre como usar o Console de gerenciamento da AWS para criar destinos de controle de utilização no nível de estágio, consulte [???](#). Para saber mais sobre como usar a AWS CLI para criar destinos de controle de utilização no nível de estágio, consulte [create-stage](#).

## Configuração de alvos de controle de utilização no nível do método em um plano de uso

É possível definir alvos de controle de utilização adicionais no nível do método em Usage Plans (Planos de uso), conforme exibido em [Criar um plano de uso](#). No console do API Gateway, isso é definido especificando Resource=<resource>, Method=<method> na configuração Configure Method Throttling (Configurar limitação de método). Por exemplo, no [caso de PetStore](#), você pode especificar Resource=/pets, Method=GET.

## APIs REST privadas no API Gateway

Uma API privada é uma API REST que só pode ser chamada de dentro de uma Amazon VPC. Você pode acessar a API usando um [endpoint de interface da VPC](#), que é uma interface de rede de endpoint que você cria na VPC. Os endpoints de interface são desenvolvidos pelo AWS PrivateLink, uma tecnologia que permite acessar de forma privada os serviços da AWS usando endereços IP privados.

Você também pode usar o Direct Connect para estabelecer uma conexão de uma rede on-premises para o Amazon VPC, depois acessar sua API privada nessa conexão. Em todos os casos, o tráfego para a API privada sempre usa conexões seguras e é isolado da internet pública. O tráfego não deixa a rede da Amazon.

### Práticas recomendadas para APIs privadas

Recomendamos que você use as seguintes práticas recomendadas ao criar uma API privada:

- Use um único endpoint da VPC para acessar várias APIs privadas. Isso reduz o número de endpoints da VPC que você pode precisar.
- Associe seu endpoint da VPC à sua API. Isso cria um registro DNS de alias do Route 53 e simplifica a invocação da API privada.
- Ative o DNS privado para sua VPC. Ao ativar o DNS privado para a VPC, é possível invocar a API em uma VPC sem transmitir o cabeçalho `Host` ou `x-apigw-api-id`.

Se você ativar a opção de DNS privado, não poderá acessar o endpoint padrão para APIs públicas. Se quiser acessar o endpoint padrão para APIs públicas, você pode desativar o DNS privado, criar uma zona hospedada privada para cada API privada na VPC e, depois, provisionar os registros necessários no Route 53. Isso permite que a API privada seja resolvida e que você, ainda assim, invoque o endpoint padrão público da VPC. Para obter mais informações, consulte [Criar uma zona hospedada privada](#).

- Restrinja o acesso à sua API privada para VPCs ou endpoints da VPC específicos. Adicione condições `aws:SourceVpc` ou `aws:SourceVpce` à política de recursos da API para restringir o acesso.
- Para obter o perímetro de dados mais seguro, você pode criar uma política de endpoint da VPC. Isso controla o acesso aos endpoints da VPC que podem invocar sua API privada.

## Considerações sobre APIs privadas

As considerações a seguir podem afetar seu uso de APIs privadas:

- Somente as APIs REST são compatíveis.
- Não é possível converter uma API privada para uma API otimizada para fronteiras.
- As APIs privadas são compatíveis somente com TLS 1.2. Versões anteriores do TLS não são compatíveis.
- Se você fizer uma solicitação usando o protocolo HTTP/2, ela será forçada a usar o protocolo HTTP/1.1.
- Não é possível definir o tipo de endereço IP para APIs privadas para permitir que somente endereços IPv4 invoquem sua API privada. Somente pilha dupla é compatível. Para obter mais informações, consulte [the section called “Tipos de endereço IP para APIs REST no API Gateway”](#).
- Para enviar tráfego usando sua API privada, é possível usar todos os tipos de endereço IP compatíveis com a Amazon VPC. Você pode enviar tráfego de pilha dupla e IPv6 configurando as definições no endpoint da VPC. Não é possível modificar isso usando o API Gateway. Para obter mais informações, consulte [Adicionar suporte a IPv6 para sua VPC](#).
- Os VPC endpoints para APIs privadas estão sujeitos às mesmas limitações dos outros VPC endpoints de interface. Para obter mais informações, consulte [Access an AWS using an interface VPC endpoint](#) (Acessar um por meio de um endpoint da VPC de interface) no AWS PrivateLink Guia. Para obter mais informações sobre como usar o API Gateway com VPCs e sub-redes compartilhadas, consulte [Sub-redes compartilhadas](#) no Guia do AWS PrivateLink.

## Próximas etapas para APIs privadas

Para saber como criar uma API privada e associar um endpoint da VPC, consulte [the section called “Criar uma API privada”](#). Para seguir um tutorial em que você cria dependências no CloudFormation e uma API privada no Console de gerenciamento da AWS, consulte [the section called “Tutorial: Crie uma API REST privada”](#).

## Criar uma API privada

Antes de criar uma API privada, crie um endpoint da VPC para o API Gateway. Depois, crie a API privada e anexe uma política de recursos a ela. Opcionalmente, você pode associar seu endpoint da VPC à API privada para simplificar a invocação da API. Por fim, implante a API.

Os procedimentos a seguir descrevem como fazer isso. Você pode criar uma API REST privada usando o Console de gerenciamento da AWS, a AWS CLI ou um AWS SDK.

## Pré-requisitos

Para seguir essas etapas, é necessário ter uma VPC totalmente configurada. Para saber como criar uma VPC, consulte [Create a VPC only](#) no Guia do usuário do Amazon VPC. Para seguir todas as etapas recomendadas ao criar uma VPC, habilite o DNS privado. Dessa forma, você pode invocar a API em uma VPC sem precisar passar o host ou o cabeçalho `x-apigw-api-id`.

Para habilitar o DNS privado, os atributos `enableDnsSupport` e `enableDnsHostnames` da sua VPC deverão ser definidos como `true`. Para obter mais informações, consulte [DNS Support in Your VPC](#) e [Updating DNS Support for Your VPC](#).

## Etapas 1: Criar um endpoint da VPC para o API Gateway em sua VPC

O procedimento a seguir mostra como criar um endpoint da VPC para o API Gateway. Para criar um endpoint da VPC para o API Gateway, especifique o domínio `execute-api` para a Região da AWS onde você está criando a API privada. O domínio `execute-api` é o serviço de componente do API Gateway para a execução da API.

Ao criar um endpoint da VPC para o API Gateway, especifique as configurações de DNS. Se você desativar o DNS privado, só poderá acessar a API usando o DNS público. Para obter mais informações, consulte [the section called “Problema: não consigo me conectar à minha API pública por um endpoint da VPC do API Gateway”](#).

## Console de gerenciamento da AWS

### Como criar um endpoint de interface da VPC para o API Gateway

1. Faça login no Console de gerenciamento da AWS e abra o console da Amazon VPC em <https://console.aws.amazon.com/vpc/>.
2. No menu de navegação, em Nuvem privada virtual, escolha Endpoints.
3. Escolha Criar endpoint.
4. (Opcional) Em Etiqueta de nome, insira um nome para ajudar a identificar o endpoint da VPC.
5. Em Categoria do serviço, escolha Serviços do AWS.
6. Em Serviços, na barra de pesquisa, insira **execute-api**. Depois, escolha o endpoint de serviço do API Gateway na Região da AWS em que você criará a API. O nome do serviço

deve ser semelhante a `com.amazonaws.us-east-1.execute-api` e o Tipo deve ser Interface.

7. Para VPC, escolha a VPC na qual você deseja criar o endpoint.
8. (Opcional) Para desativar a opção Habilitar nome DNS privado, escolha Configurações adicionais e desmarque a opção Habilitar nome DNS privado.
9. Em Sub-redes, selecione as zonas de disponibilidade em que criou as interfaces de rede do endpoint. Para melhorar a disponibilidade da sua API, escolha várias sub-redes.
10. Em Security group (Grupo de segurança), selecione os grupos de segurança a serem associados às interfaces de rede do VPC endpoint.

O security group que você escolher deve ser definido para permitir o tráfego de entrada de HTTPS na porta TCP 443 a partir de um intervalo de IP na sua VPC ou de outro security group na sua VPC.

11. Em Política, siga um destes procedimentos:
  - Se você não criou a API privada ou não quer configurar uma política de endpoint da VPC personalizada, escolha Acesso total.
  - Se você já criou uma API privada e quer configurar uma política de endpoint da VPC personalizada, é possível inserir uma política de endpoint da VPC personalizada. Para obter mais informações, consulte [the section called “Usar políticas de VPC endpoint para APIs privadas”](#).

Você pode atualizar a política de endpoint da VPC depois de criar o endpoint da VPC. Para obter mais informações, consulte [Update a VPC endpoint policy](#).

12. Escolha Criar endpoint.
13. Copie o ID de endpoint da VPC resultante, pois talvez precise usá-lo em etapas futuras.

## AWS CLI

O comando [create-vpc-endpoint](#) indicado abaixo pode ser usado para criar um endpoint da 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 o ID de endpoint da VPC resultante, pois talvez precise usá-lo em etapas futuras.

## Etapa 2: Crie uma API privada

Depois de criar o endpoint da VPC, crie uma API REST privada. O procedimento a seguir mostra como criar uma API privada.

### Console de gerenciamento da AWS

Para criar uma API privada

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione Create API (Criar API).
3. Em REST API, escolha Build (Criar).
4. Em Nome, insira um nome.
5. (Opcional) Em Description (Descrição), insira uma descrição.
6. Em Tipo de endpoint de API, escolha Privado.
7. (Opcional) Em ID do endpoint da VPC, insira um ID de endpoint da VPC.

Se você associar um ID de endpoint da VPC à sua API privada, poderá invocar a API de dentro da VPC sem substituir um cabeçalho Host ou transmitir um x-apigw-api-id header. Para obter mais informações, consulte [the section called “\(Opcional\) Associar ou desassociar um endpoint da VPC a uma API privada”](#).

8. Em Tipo de endereço IP, escolha Dualstack.
9. Selecione Criar API.

Depois de concluir as etapas anteriores, você poderá seguir as mesmas instruções em [the section called “Começar a usar o console da API REST”](#) para configurar métodos e integrações para essa API, mas não poderá implantar a API. Para implantar a API, siga a etapa 3 e anexe uma política de recursos à API.

### AWS CLI

O comando [create-rest-api](#) indicado abaixo cria uma 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" }'
```

Uma chamada bem-sucedida retorna uma saída semelhante ao seguinte:

```
{
 "createdDate": "2017-10-13T18:41:39Z",
 "description": "Simple private PetStore API",
 "endpointConfiguration": {
 "types": [
 "PRIVATE"
],
 "ipAddressType": "dualstack"
 },
 "id": "0qzs2sy7bh",
 "name": "Simple PetStore (AWS CLI, Private)"
}
```

Depois de concluir as etapas anteriores, você poderá seguir as mesmas instruções em [the section called “Tutorial: como criar uma API REST usando SDKs da AWS ou a AWS CLI”](#) para configurar métodos e integrações para essa API, mas não poderá implantar a API. Para implantar a API, siga a etapa 3 e anexe uma política de recursos à API.

## SDK JavaScript v3

O seguinte exemplo mostra como criar uma API privada usando o 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)
}
};
```

Uma chamada bem-sucedida retorna uma saída semelhante ao seguinte:

```
{
 apiKeySource: 'HEADER',
 createdAt: 2024-04-03T17:56:36.000Z,
 description: 'Demo private API created using the AWS SDK for JavaScript v3',
 disableExecuteApiEndpoint: false,
 endpointConfiguration: { types: ['PRIVATE'] },
 id: 'abcd1234',
 name: 'Simple PetStore (JavaScript v3 SDK, private)',
 rootResourceId: 'efg567',
 version: '0.00.001'
}
```

Depois de concluir as etapas anteriores, você poderá seguir as mesmas instruções em [the section called “Tutorial: como criar uma API REST usando SDKs da AWS ou a AWS CLI”](#) para configurar métodos e integrações para essa API, mas não poderá implantar a API. Para implantar a API, siga a etapa 3 e anexe uma política de recursos à API.

## Python SDK

O seguinte exemplo mostra como criar uma API privada usando o 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

```

Uma chamada bem-sucedida retorna uma saída semelhante ao seguinte:

```

"{\"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\"}"

```

Depois de concluir as etapas anteriores, você poderá seguir as mesmas instruções em [the section called “Tutorial: como criar uma API REST usando SDKs da AWS ou a AWS CLI”](#) para configurar métodos e integrações para essa API, mas não poderá implantar a API. Para implantar a API, siga a etapa 3 e anexe uma política de recursos à API.

### Etapla 3: Configurar uma política de recursos para uma API privada

Sua API privada atual está inacessível para todas as VPCs. Use uma política de recursos para conceder às suas VPCs e endpoints da VPC acesso às suas APIs privadas. É possível conceder acesso a um endpoint da VPC em qualquer conta da AWS.

Sua política de recursos deve conter as condições `aws:SourceVpc` ou `aws:SourceVpce` para restringir o acesso. Recomendamos que você identifique VPCs e endpoints da VPC específicos e não crie uma política de recursos que permita acesso a todas as VPCs e endpoints da VPC.

O procedimento a seguir mostra como anexar uma política de recursos à sua API.

## Console de gerenciamento da AWS

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API REST.
3. No painel de navegação principal, escolha Política de recursos.
4. Escolha Criar política.
5. Escolha Seleccionar um modelo e selecione Source VPC (VPC de origem).
6. Substitua `{{vpcID}}` (incluindo as chaves) pelo ID da VPC.
7. Escolha Save changes.

## AWS CLI

O comando [update-rest-api](#) indicado abaixo anexa uma política de recursos a uma API existente:

```
aws apigateway update-rest-api \
 --rest-api-id a1b2c3 \
 --patch-operations op=replace,path=/
policy,value='{"jsonEscapedPolicyDocument"}'
```

Talvez você também queira controlar quais recursos têm acesso ao seu endpoint da VPC. Para controlar quais recursos têm acesso ao endpoint da VPC, anexe uma política de endpoint ao endpoint da VPC. Para obter mais informações, consulte [the section called “Usar políticas de VPC endpoint para APIs privadas”](#).

(Opcional) Associar ou desassociar um endpoint da VPC a uma API privada

Quando você associa um endpoint da VPC a uma API privada, o API Gateway gera um registro DNS de alias do Route 53. Você pode usar esse registro para invocar suas APIs privadas da mesma forma como o faz para suas APIs públicas sem substituir um cabeçalho Host ou transmitir um cabeçalho `x-apigw-api-id`.

O URL base gerado está no seguinte formato:

```
https://{rest-api-id}-{vpce-id}.execute-api.{region}.amazonaws.com/{stage}
```

## Associate a VPC endpoint (Console de gerenciamento da AWS)

É possível associar um endpoint da VPC a uma API privada tanto durante a criação quanto depois. O procedimento a seguir mostra como associar um endpoint da VPC a uma API já criada.

Como associar um endpoint da VPC a uma API privada

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione a API privada.
3. No painel de navegação principal, escolha Política de recursos.
4. Edite a política de recursos para permitir chamadas do endpoint da VPC adicional.
5. No painel de navegação principal, selecione Configurações da API.
6. Na seção Detalhes da API, escolha Editar.
7. Em IDs de endpoint da VPC, selecione IDs de endpoint da VPC adicionais.
8. Escolha Salvar.
9. Implante a API novamente para que as alterações entrem em vigor.

## Dissociate a VPC endpoint (Console de gerenciamento da AWS)

Como dissociar um endpoint da VPC de uma API REST privada

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione a API privada.
3. No painel de navegação principal, escolha Política de recursos.
4. Edite a política de recursos para remover as menções ao endpoint da VPC que você deseja dissociar da API privada.
5. No painel de navegação principal, selecione Configurações da API.
6. Na seção Detalhes da API, escolha Editar.
7. Em IDs de endpoint da VPC, clique no X para dissociar o endpoint da VPC.
8. Escolha Salvar.
9. Implante a API novamente para que as alterações entrem em vigor.

## Associate a VPC endpoint (AWS CLI)

O comando [create-rest-api](#) indicado abaixo associa endpoints da VPC no momento da criação da API:

```
aws apigateway create-rest-api \
 --name Petstore \
 --endpoint-configuration '{ "types": ["PRIVATE"], "vpcEndpointIds" :
 ["vpce-0212a4ababd5b8c3e", "vpce-0393a628149c867ee"] }' \
 --region us-west-2
```

A saída será exibida da seguinte forma:

```
{
 "apiKeySource": "HEADER",
 "endpointConfiguration": {
 "types": [
 "PRIVATE"
],
 "vpcEndpointIds": [
 "vpce-0212a4ababd5b8c3e",
 "vpce-0393a628149c867ee"
]
 },
 "id": "u67n3ov968",
 "createdDate": 1565718256,
 "name": "Petstore"
}
```

O comando [update-rest-api](#) indicado abaixo associa endpoints da VPC a uma API já criada:

```
aws apigateway update-rest-api \
 --rest-api-id u67n3ov968 \
 --patch-operations "op='add',path='/endpointConfiguration/
vpcEndpointIds',value='vpce-01d622316a7df47f9'" \
 --region us-west-2
```

A saída será exibida da seguinte forma:

```
{
```

```

 "name": "Petstore",
 "apiKeySource": "1565718256",
 "tags": {},
 "createdDate": 1565718256,
 "endpointConfiguration": {
 "vpcEndpointIds": [
 "vpce-0212a4ababd5b8c3e",
 "vpce-0393a628149c867ee",
 "vpce-01d622316a7df47f9"
],
 "types": [
 "PRIVATE"
]
 },
 "id": "u67n3ov968"
 }
}

```

Implante a API novamente para que as alterações entrem em vigor.

Disassociate a VPC endpoint (AWS CLI)

O comando [update-rest-api](#) indicado abaixo desassocia um endpoint da VPC de uma 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

```

A saída será exibida da seguinte forma:

```

{
 "name": "Petstore",
 "apiKeySource": "1565718256",
 "tags": {},
 "createdDate": 1565718256,
 "endpointConfiguration": {
 "vpcEndpointIds": [
 "vpce-0212a4ababd5b8c3e",
 "vpce-01d622316a7df47f9"
],
 "types": [
 "PRIVATE"
]
 }
}

```

```
]
 },
 "id": "u67n3ov968"
}
```

Implante a API novamente para que as alterações entrem em vigor.

#### Etapa 4: Implantar uma API privada

Para implantar uma API, crie uma implantação de API e a associe a um estágio. O procedimento a seguir mostra como implantar uma API privada.

#### Console de gerenciamento da AWS

##### Como implantar uma API privada

1. Selecione a API.
2. Escolha Implantar API.
3. Em Estágio, selecione Novo estágio.
4. Em Nome do estágio, insira o nome de um estágio.
5. (Opcional) Em Description (Descrição), insira uma descrição.
6. Escolha Implantar.

#### AWS CLI

O comando [create-deployment](#) indicado abaixo implanta uma API privada:

```
aws apigateway create-deployment --rest-api-id a1b2c3 \
 --stage-name test \
 --stage-description 'Private API test stage' \
 --description 'First deployment'
```

#### Solucionar problemas da API privada

O tópico a seguir fornece orientações para a solução de erros e problemas que você pode encontrar ao criar uma API privada.

Problema: não consigo me conectar à minha API pública por um endpoint da VPC do API Gateway

Ao criar uma VPC, você pode definir as configurações de DNS. Recomendamos ativar a opção de DNS privado para a VPC. Se você optar por não ativar o DNS privado, só poderá acessar a API via DNS público.

Se você habilitar o DNS privado, não poderá acessar o endpoint padrão de uma API pública do API Gateway pelo endpoint da VPC. É possível acessar uma API com um nome de domínio personalizado.

Se você criar um nome de domínio personalizado regional, use um registro de alias do tipo A; se você criar um nome de domínio personalizado otimizado para bordas, não haverá restrições para o tipo de registro. Você pode acessar essas APIs públicas com o DNS privado habilitado. Para obter mais informações, consulte [Issue: I connect to my public API from an API Gateway VPC endpoint](#).

Problema: minha API retorna **{"Message":"User: anonymous is not authorized to perform: execute-api:Invoke on resource: arn:aws:execute-api:us-east-1:\*\*\*\*\*/\*\*\*\*/\*\*\*\*/"}**

Em sua política de recursos, se você definir a entidade principal como uma entidade principal da AWS, como a seguinte:

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"
]
 }
]
}
```



```
}
```

Você deverá usar a autorização `AWS_IAM` para todos os métodos em sua API, caso contrário, a API retornará a mensagem de erro anterior. Para obter mais instruções sobre como ativar a autorização `AWS_IAM` para um método, consulte [the section called “Métodos”](#).

Problema: não consigo saber se meu endpoint da VPC está associado à minha API

Se você associar ou desassociar um endpoint da VPC de sua API privada, será necessário reimplantar a API. A operação de atualização pode levar alguns minutos para ser concluída devido à propagação do DNS. Durante esse período, sua API está disponível, mas a propagação de DNS para os URLs do DNS recém-gerados ainda poderá estar em andamento. Se os novos URLs não estiverem sendo resolvidos no DNS depois de alguns minutos, recomendamos que você reimplante a API.

## Nomes de domínio personalizados para APIs privadas no API Gateway

Também é possível criar um nome de domínio personalizado para APIs privadas. Use um nome de domínio personalizado privado para fornecer aos chamadores de API um URL mais simples e intuitivo. Com um nome de domínio personalizado privado, você pode reduzir a complexidade, configurar medidas de segurança durante o handshake do TLS e controlar o ciclo de vida do certificado do seu nome de domínio usando o AWS Certificate Manager (ACM). Para obter mais informações, consulte [the section called “Proteger a chave privada do certificado para o nome de domínio personalizado”](#).

Os nomes de domínio personalizados para APIs privadas não precisam ser exclusivos em várias contas. Você pode criar `example.private.com` na conta 111.122.223.333 e na conta 555.555.555.555, desde que seu certificado do ACM abranja o nome de domínio. Para identificar um nome de domínio personalizado privado, use o respectivo ARN. Esse identificador é exclusivo para nomes de domínio personalizados privados.

Quando você cria um nome de domínio personalizado privado no API Gateway, você é um provedor de API. Você pode fornecer seu nome de domínio personalizado privado a outras Contas da AWS usando o API Gateway ou o AWS Resource Access Manager (AWS RAM).

Quando você invoca um nome de domínio personalizado privado, você é um consumidor de API. Você pode consumir um nome de domínio personalizado privado da sua Conta da AWS ou de outra Conta da AWS.

Ao consumir um nome de domínio personalizado privado, você cria uma associação de acesso ao nome de domínio entre um endpoint da VPC e um nome de domínio personalizado privado. Com uma associação de acesso ao nome de domínio, os consumidores de API podem invocar seu nome de domínio personalizado privado enquanto estão isolados da internet pública. Para obter mais informações, consulte [the section called “Provedores e consumidores de API”](#).

Proteger a chave privada do certificado para o nome de domínio personalizado

Quando você solicita um certificado SSL/TLS usando o ACM para criar um nome de domínio personalizado para APIs privadas, o ACM gera um par de chaves públicas/privadas. Ao importar um certificado, você gera o par de chaves. A chave pública se torna parte do certificado. Para armazenar com segurança a chave privada, o ACM cria outra chave usando o AWS KMS, chamada de chave do KMS, com o alias `aws/acm`. O AWS KMS usa essa chave para criptografar a chave privada do seu certificado. Para obter mais informações, consulte [Proteção de dados no AWS Certificate Manager](#), no Guia do usuário do AWS Certificate Manager.

O API Gateway usa o AWS TLS Connection Manager, um serviço que só pode ser acessado por Serviços da AWS, para proteger e usar as chaves privadas do seu certificado. Quando você usa seu certificado do ACM para criar um nome de domínio personalizado do API Gateway, o API Gateway associa seu certificado ao AWS TLS Connection Manager. Fazemos isso criando uma concessão no AWS KMS relação à sua chave gerenciada pela AWS. Essa concessão permite que o TLS Connection Manager use o AWS KMS para descriptografar a chave privada do seu certificado. O TLS Connection Manager usa o certificado e a chave privada descriptografada (texto simples) para estabelecer uma conexão segura (sessão SSL/TLS) com clientes de serviços do API Gateway. Quando o certificado for desassociado de um serviço do API Gateway, a concessão será removida. Para obter mais informações, consulte [Concessões](#) no Guia do desenvolvedor do AWS Key Management Service.

Para obter mais informações, consulte [the section called “Criptografia em repouso”](#).

Considerações referentes a nomes de domínio personalizados privados

As considerações a seguir podem afetar o uso de nomes de domínio personalizados privados.

- O API Gateway leva em torno de 15 minutos para provisionar seu nome de domínio personalizado privado.
- Se você atualizar seu certificado do ACM, o API Gateway levará cerca de 15 minutos para concluir a atualização. Durante esse período, seu nome de domínio fica no estado `UPDATING`, mas você ainda pode acessá-lo.

- Para invocar um nome de domínio personalizado privado, você deve criar uma associação de acesso ao nome de domínio. Depois que uma associação de acesso ao nome de domínio é criada, ela leva cerca de 15 minutos para ficar pronta.
- O ARN do nome de domínio personalizado privado contém o *account-id* e o *domain-name-id*. Quando você cria um nome de domínio, o API Gateway usa o formato de ARN `arn:partition:apigateway:region::/domainnames/domain-name`. Ao acessar um nome personalizado de domínio personalizado, use o formato de ARN `arn:partition:apigateway:region:account-id:/domainnames/domain-name+domain-name-id`.

Talvez seja necessário modificar as permissões do IAM para autorizar o acesso a um nome de domínio privado depois de criá-lo.

- Não é possível invocar nomes de domínio personalizados privados usando o mesmo nome de um mesmo endpoint da VPC. Por exemplo, se você quiser invocar `arn:aws:apigateway:us-west-2:111122223333:/domainnames/private.example.com+abcd1234` e `arn:aws:apigateway:us-west-2:111122223333:/domainnames/private.example.com+xyz000`, associe cada nome de domínio personalizado privado a um endpoint da VPC diferente.
- Certificados curinga são permitidos, como um certificado para `*.private.example.com`.
- Nomes de domínio personalizados curinga não são permitidos.
- Somente certificados RSA com um tamanho de chave de 2.048 bits e certificados ECDSA com um tamanho de chave de 256 e 384 bits são permitidos.
- Não é possível definir o tipo de endereço IP para APIs privadas para permitir que somente endereços IPv4 invoquem sua API privada. Somente pilha dupla é compatível. Para obter mais informações, consulte [the section called “Tipos de endereço IP para APIs REST no API Gateway”](#).
- Para enviar tráfego usando sua API privada, é possível usar todos os tipos de endereço IP compatíveis com a Amazon VPC. Você pode enviar tráfego de pilha dupla e IPv6 configurando as definições no endpoint da VPC. Não é possível modificar isso usando o API Gateway. Para obter mais informações, consulte [Adicionar suporte a IPv6 para sua VPC](#).
- O mapeamento de caminho base de vários níveis, como associar a API privada a `/developers/feature`, não é permitido, mas é possível usar uma regra de roteamento para criar uma condição de caminho de vários níveis. Para obter mais informações, consulte [the section called “Encaminhar o tráfego às APIs por meio do nome de domínio personalizado no API Gateway”](#).
- Não é possível definir uma versão mínima do TLS para o nome de domínio personalizado privado. Todos os nomes de domínio personalizados privados têm a política de segurança TLS-1-2.

- Você pode usar a política de endpoint da VPC para controlar o acesso a um nome de domínio personalizado privado. Para ter mais informações, consulte 4 e 5 em [the section called “Usar políticas de VPC endpoint para APIs privadas”](#).
- Você deve criar uma política de recursos separada para a API privada e o nome de domínio personalizado privado. Para invocar um nome de domínio personalizado privado, um consumidor de API precisa de acesso à política de recursos do nome de domínio personalizado privado, à política de recursos da API privada e a quaisquer políticas de endpoint da VPC ou autorizações na API privada.

## Considerações sobre o uso de nomes de domínio personalizados privados com outros recursos do API Gateway

As seguintes considerações podem afetar a forma como você usa nomes de domínio personalizados privados com outros recursos do API Gateway:

- Não é possível enviar tráfego de um nome de domínio personalizado privado a uma API pública.
- Quando uma API privada é associada a um nome de domínio personalizado privado, não é possível alterar o tipo de endpoint da API.
- Não é possível migrar um nome de domínio personalizado público para um nome de domínio personalizado privado.
- Se você usa um endpoint da VPC para acessar um nome de domínio personalizado público, não o use para criar uma associação de acesso ao nome de domínio com um nome de domínio personalizado privado.

## Diferenças entre nomes de domínio personalizados privados e nomes de domínio personalizados públicos

A seguir é apresentada uma descrição das diferenças entre nomes de domínio personalizados privados e públicos:

- Os nomes de domínio personalizados privados não precisam ser exclusivos em várias contas.
- Um nome de domínio privado tem um ID de nome de domínio. Esse ID distingue exclusivamente um nome de domínio personalizado privado e não é gerado para nomes de domínio personalizados públicos.
- Ao usar a AWS CLI para atualizar ou excluir um nome de domínio personalizado privado, você deve fornecer o respectivo ID. Se você tiver um nome de domínio personalizado privado

chamado `example.com` e um nome de domínio personalizado público chamado `example.com` e não fornecer o respectivo ID, o API Gateway modificará ou excluirá seu nome de domínio personalizado público.

Próximas etapas referentes a nomes de domínio personalizados para APIs privadas

Para ter informações sobre as tarefas de um provedor de API e de um consumidor de API, consulte [the section called “Provedores e consumidores de API”](#).

Para obter instruções sobre como criar um nome de domínio personalizado privado que você pode invocar em sua Conta da AWS, consulte [the section called “Tutorial: como criar e invocar um nome de domínio personalizado para APIs privadas”](#).

Para obter instruções sobre como fornecer a outra Conta da AWS acesso ao seu nome de domínio personalizado privado, consulte [the section called “Provedor de API: compartilhar seu nome de domínio personalizado privado usando o AWS RAM”](#). Para obter instruções sobre como associar o endpoint da VPC a um nome de domínio personalizado privado em outra Conta da AWS, consulte [the section called “Consumidor de API: associar o endpoint da VPC a um nome de domínio personalizado privado compartilhado com você”](#).

Tarefas dos provedores e consumidores de API referentes a nomes de domínio personalizados para APIs privadas

Quando você cria um nome de domínio personalizado privado, você é um provedor de API. Quando você invoca um nome de domínio personalizado privado, você é um consumidor de API. Você pode consumir um nome de domínio personalizado privado da sua Conta da AWS ou de outra Conta da AWS.

A seção a seguir explica as tarefas exigidas pelo provedor e pelo consumidor de API para usar um nome de domínio personalizado privado. Quando você invoca um nome de domínio personalizado privado na sua Conta da AWS, você é tanto provedor quanto consumidor de API. Se quiser invocar um domínio personalizado privado em outra Conta da AWS, dependendo da relação de confiança entre o provedor de API e o consumidor de API no AWS Organizations, o AWS RAM poderá concluir algumas tarefas para você.

Tarefas de um provedor de API

Os provedores de API criam APIs privadas e as associam a nomes de domínio personalizados.

Eles gerenciam duas políticas de recursos para proteger seus nomes de domínio personalizados privados. A primeira política, destinada ao serviço `execute-api`, controla quais endpoints da VPC podem invocar seu nome de domínio personalizado privado. Na configuração de nome de domínio personalizado privado, ela é chamada de `policy`.

A segunda política, destinada ao serviço Amazon API Gateway Management, controla quais endpoints da VPC em outras Contas da AWS podem formar uma associação de acesso ao nome de domínio com seu nome de domínio personalizado privado. Um endpoint da VPC precisa formar uma associação de acesso ao nome de domínio com um nome de domínio personalizado privado para invocá-lo. Na configuração de nome de domínio personalizado privado, trata-se da `managementPolicy`. Você pode usar o AWS RAM ou o API Gateway para atualizar essa política. Se você não pretende permitir que endpoints da VPC em outras Contas da AWS invoquem seu nome de domínio personalizado, não edite a `managementPolicy`.

Se você for provedor de API, deverá fazer o seguinte:

1. Crie uma API privada.
2. Atualize a `policy` da API privada para conceder ao seu endpoint da VPC acesso à API privada.
3. Crie um nome de domínio personalizado.
4. Atualize a `policy` do nome de domínio personalizado privado para conceder ao seu endpoint da VPC acesso ao nome de domínio personalizado privado.
5. Crie um mapeamento de caminho base ou uma regra de roteamento para enviar tráfego da API privada ao nome de domínio personalizado privado. Para obter mais informações, consulte [the section called “Encaminhar o tráfego às APIs por meio do nome de domínio personalizado no API Gateway”](#).

Se você quiser permitir que consumidores de API em outras Contas da AWS acessem seu nome de domínio personalizado privado, faça o seguinte:

1. Atualize a `managementPolicy` do nome de domínio personalizado privado para permitir que consumidores de API em outras contas associem o endpoint da VPC deles ao seu nome de domínio personalizado privado. Isso pode ser feito usando um dos métodos a seguir:

#### AWS RAM

Com o AWS RAM, se o provedor de API e o consumidor de API estiverem na mesma organização que está usando o AWS Organizations, o compartilhamento de recursos entre o provedor e o consumidor será aceito automaticamente. Do contrário, você deverá esperar até

que o consumidor de API aceite o compartilhamento de recursos. Recomendamos que você use o AWS RAM para compartilhar seu nome de domínio personalizado privado.

## API Gateway

Com o API Gateway, é possível usar somente a AWS CLI. Você deve atualizar seu nome de domínio personalizado privado usando uma operação de patch e fornecer seu próprio documento de política para a `managementPolicy`.

2. Atualize a `policy` do nome de domínio personalizado privado e quaisquer APIs privadas associadas a ele para conceder acesso ao endpoint da VPC do consumidor de API.

Para obter instruções sobre como fornecer sua API a outra Conta da AWS, consulte [the section called “Provedor de API: compartilhar seu nome de domínio personalizado privado usando o AWS RAM”](#).

## Tarefas de um consumidor de API

Os consumidores de API associam seus endpoints da VPC a um ARN de nome de domínio para invocar um nome de domínio personalizado privado. Os consumidores de API não precisam criar uma API do API Gateway.

Se você for um consumidor de API, faça o seguinte:

1. Crie um endpoint da VPC com DNS privado habilitado na Amazon VPC.
2. (Opcional, se for usado o AWS RAM) Aceite um compartilhamento de recursos de domínio personalizado privado no AWS RAM dentro de 12 horas após o compartilhamento do recurso. Se você e o provedor de API estiverem na mesma organização, o compartilhamento de recursos será aceito automaticamente.
3. Obtenha o ARN do nome de domínio personalizado privado. Como o URL do nome de domínio personalizado privado não é exclusivo, você usa o respectivo ARN para formar a associação de acesso ao nome de domínio entre o endpoint da VPC e o nome de domínio personalizado privado. Você pode usar o AWS RAM para recuperar o ARN do nome de domínio personalizado privado.
4. Associe o ARN do domínio personalizado privado ao seu endpoint da VPC no API Gateway. Isso cria uma conexão segura entre o endpoint da VPC e o nome de domínio personalizado privado. O tráfego não deixa a rede da Amazon.
5. Aguarde até que o provedor de API conceda ao seu endpoint da VPC acesso ao nome de domínio personalizado privado e a todas as APIs privadas associadas ao nome de domínio personalizado



privado. Se você for provedor e consumidor de API, conceda acesso de invocação ao seu próprio endpoint da VPC.

6. Crie uma zona hospedada privada do Route 53 e um registro do Route 53 para resolver o nome de domínio personalizado privado no Route 53.

Para obter instruções sobre como consumir uma API em outra Conta da AWS, consulte [the section called “Consumidor de API: associar o endpoint da VPC a um nome de domínio personalizado privado compartilhado com você”](#).

Tutorial: como criar e invocar um nome de domínio personalizado para APIs privadas

Neste tutorial, você cria um nome de domínio personalizado privado que pode ser invocado em uma VPC em sua conta. Para conseguir isso, você é o provedor e o consumidor da API. Você precisa de uma API privada e de um endpoint da VPC existentes para concluir este tutorial. Se você usa um endpoint da VPC para acessar um nome de domínio personalizado público, não o use para este tutorial ou para criar associações de acesso a nomes de domínio.

Etapa 1: criar um nome de domínio personalizado

Para criar um nome de domínio personalizado privado, especifique o nome de domínio, o certificado do ACM e a política do serviço `execute-api` para controlar quais endpoints da VPC podem invocá-lo.

Console de gerenciamento da AWS

Como criar um nome de domínio personalizado

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, selecione Nomes de domínio personalizados.
3. Escolha Adicionar nome do domínio.
4. Em Nome de domínio, insira um nome de domínio

Seu certificado do ACM deve abranger esse nome de domínio, mas o nome de domínio não precisa ser exclusivo.

5. Selecione Privado.
6. Em Modo de roteamento, escolha Somente mapeamentos de API.
7. Em Certificado do ACM, escolha um certificado.



## 8. Escolha Adicionar nome do domínio.

O API Gateway provisiona um nome de domínio com uma política deny que nega acesso a todos os recursos. Essa é a política de recursos do serviço `execute-api`. Você precisa atualizar essa política de recursos para conceder acesso aos seus endpoints da VPC e permitir que eles invoquem seu nome de domínio personalizado privado.

Como atualizar sua política de recursos

1. Escolha a guia Política de recursos e selecione Editar política de recursos.
2. Insira a política de recursos a seguir no editor de código. Substitua o endpoint da VPC `vpce-abcd1234efg` pelo seu ID de endpoint da 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. Escolha Salvar alterações.

## AWS CLI

Ao criar um nome de domínio personalizado privado usando a AWS CLI, você fornece uma política de recursos para o serviço `execute-api` conceder acesso aos endpoints da VPC e permitir que eles invoquem seu nome de domínio personalizado privado por meio do parâmetro `--policy file://policy.json`. Você pode modificar essa política posteriormente.

Neste exemplo, você anexará a política de recursos a seguir como `policy` carregando os parâmetros de um arquivo. Copie e salve esse arquivo como `policy.json`. Essa política permite apenas tráfego de entrada para um nome de domínio personalizado privado originado do endpoint da 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"
 }
 }
 }
]
}
```

O comando [create-domain-name](#) indicado abaixo cria um nome de domínio 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
```

A saída será exibida da seguinte forma:

```
{
 "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": "..."
}
```

Etapa 2: criar um mapeamento de caminho base para associar a API privada ao nome de domínio personalizado privado

Depois de criar um nome de domínio personalizado privado, você associa uma API privada a ele. O mapeamento de caminho base torna uma API acessível por meio da combinação do nome de domínio personalizado e do caminho base correspondente. Recomendamos usar um único nome de domínio personalizado privado como o nome do host de várias APIs privadas.

Todos os provedores de API precisam criar um mapeamento de caminho base, mesmo que você não pretenda invocar sua API. Também é necessário conceder acesso para que os endpoints da VPC invoquem quaisquer APIs privadas que você associa ao nome de domínio personalizado privado.

## Console de gerenciamento da AWS

### Como criar um mapeamento de caminho base

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, selecione Nomes de domínio personalizados.
3. Escolha um nome de domínio personalizado privado.
4. Na guia Mapeamentos de API, escolha Configurar mapeamentos.
5. Escolha Add new mapping (Adicionar novo mapeamento).
6. Insira uma API, um Estágio e, opcionalmente, um Caminho.
7. Escolha Salvar.

## AWS CLI

O comando [create-base-path-mapping](#) indicado abaixo criar um mapeamento entre uma API privada e um nome de domínio 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
```

A saída será exibida da seguinte forma:

```
{
 "basePath": "v1",
 "restApiId": "a1b2c3",
 "stage": "prod"
}
```

Para ter maior flexibilidade em relação a como você direciona o tráfego para as APIs, você pode alterar o modo de roteamento para `ROUTING_RULE_ONLY` ou `ROUTING_RULE_THEN_API_MAPPING`

e criar uma regra de roteamento. Para obter mais informações, consulte [the section called “Encaminhar o tráfego às APIs por meio do nome de domínio personalizado no API Gateway”](#).

 Note

Se você quiser que outras Contas da AWS invoquem seu nome de domínio personalizado privado, depois de concluir este tutorial, siga as etapas em [the section called “Provedor de API: compartilhar seu nome de domínio personalizado privado usando o AWS RAM”](#).

Etapa 3: criar uma associação de acesso ao nome de domínio entre o nome de domínio personalizado e um endpoint da VPC

Em seguida, você cria uma associação de acesso ao nome de domínio entre o nome de domínio personalizado privado e o endpoint da VPC. Seu endpoint da VPC usa a associação de acesso ao nome de domínio para invocar o nome de domínio personalizado privado enquanto está isolado da internet pública.

## Console de gerenciamento da AWS

Como criar uma associação de acesso ao nome de domínio

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, selecione Nomes de domínio personalizados.
3. Escolha um nome de domínio personalizado privado.
4. Na guia Compartilhamento de recursos, em Associações de acesso a nomes de domínio, escolha Criar associação de acesso a nomes de domínio.
5. Em ARN do nome do domínio, selecione seu nome de domínio.
6. Em IDs de endpoint da VPC, selecione o ID de endpoint da VPC ao qual você forneceu acesso na Etapa 1.
7. Escolha Associação de acesso a nomes de domínio.

Você também pode criar sua associação de acesso ao nome de domínio usando a página Associações de acesso a nomes de domínio do console.

## AWS CLI

O comando `create-domain-name-access-association` a seguir cria uma associação de acesso ao nome de domínio entre o nome de domínio personalizado privado e o endpoint da 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
```

A saída será exibida da seguinte forma:

```
{
 "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"
}
```

Depois que a associação de acesso ao nome de domínio é criada, ela leva cerca de 15 minutos para ficar pronta. Enquanto isso, você pode prosseguir com as etapas a seguir.

### Etapa 4: criar uma zona hospedada do Route 53

Depois de atualizar sua política de recursos e associar o nome de domínio personalizado privado ao endpoint da VPC, você cria uma zona hospedada privada no Route 53 para resolver o nome de domínio personalizado. Zona hospedada é um contêiner que abriga informações sobre como você deseja rotear o tráfego para um domínio dentro de uma ou mais VPCs sem expor seus recursos à internet. Para obter mais informações, consulte [Como trabalhar com zonas hospedadas privadas](#).

### Console de gerenciamento da AWS

Para usar o Console de gerenciamento da AWS, consulte [Creating a private hosted zone](#) no Guia do desenvolvedor do Amazon Route 53.

Em Nome, use o nome do seu domínio personalizado privado. Em ID da VPC, use a VPC que contém o endpoint da VPC que você usou nas etapas anteriores.

## AWS CLI

O comando [create-hosted-zone](#) indicado abaixo cria uma zona hospedada 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
```

A saída contém o ID da zona hospedada. Você usa o ID da zona hospedada nas etapas a seguir.

## Etapas 5: criar um registro de DNS do Route 53

Depois de criar a zona hospedada, crie um registro para resolver o nome de domínio personalizado privado. Use o ID da zona hospedada criado na etapa anterior. Neste exemplo, você cria um tipo de registro A. Se estiver usando IPv6 para o endpoint da VPC, crie um tipo de registro AAAA. Se estiver usando pilha dupla para o endpoint da VPC, crie um registro do tipo AAAA e um do tipo A.

## Console de gerenciamento da AWS

Para usar o Console de gerenciamento da AWS, consulte [Routing traffic to an Amazon API Gateway API by using your domain name](#).

Use Criação rápida e ative Alias. Para o endpoint, use o nome DNS do endpoint da VPC.

## AWS CLI

Para configurar os registros de DNS para associar o nome de domínio personalizado privado ao nome de host do ID de zona hospedada fornecido, crie um arquivo JSON que contenha a configuração de um registro de DNS para o nome de domínio privado.

O comando `setup-dns-record.json` a seguir mostra como criar um registro de DNS A para associar um nome de domínio personalizado privado ao respectivo nome de host privado. Você fornece o DNSName do ID do DNS da VPC e o ID da zona hospedada que você criou na etapa anterior.

```
{
 "Changes": [
 {
```

```
"Action": "UPSERT",
"ResourceRecordSet": {
 "Name": "private.example.com",
 "Type": "A",
 "AliasTarget": {
 "DNSName": "vpce-abcd1234.execute-api.us-west-2.vpce.amazonaws.com",
 "HostedZoneId": "Z20JLYMU09EFXC",
 "EvaluateTargetHealth": false
 }
}
```

O comando [change-resource-record-sets](#) indicado abaixo cria um registro de DNS para o nome de domínio personalizado privado:

```
aws route53 change-resource-record-sets \
 --hosted-zone-id ZABCDEFGHI1234 \
 --change-batch file://path/to/your/setup-dns-record.json
```

Substitua `hosted-zone-id` pelo ID da zona hospedada do Route 53 do conjunto de registros DNS na conta. O valor do parâmetro `change-batch` aponta para um arquivo JSON.

Caso não pretenda invocar seu nome de domínio personalizado privado, depois de confirmar que ele está funcionando, você poderá excluir esses recursos.

#### Etapa 6: invocar o nome de domínio personalizado privado

Agora você pode invocar o nome de domínio personalizado privado em sua Conta da AWS. Em sua VPC, use o comando `curl` a seguir para acessar o nome de domínio personalizado privado.

```
curl https://private.example.com/v1
```

Para ter mais informações sobre outras formas de invocar sua API privada, consulte [the section called “Invocar uma API privada usando um nome de domínio personalizado”](#).

#### Etapa 7: limpar

Para evitar custos desnecessários, exclua a associação entre o endpoint da VPC e o nome de domínio personalizado privado e, em seguida, exclua o nome de domínio personalizado privado.



## Console de gerenciamento da AWS

Como excluir a associação de acesso ao nome de domínio

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, escolha Associações de acesso a nomes de domínio.
3. Selecione sua associação de acesso ao nome de domínio e escolha Excluir.
4. Confirme sua escolha e selecione Excluir.

Depois de excluir sua associação de acesso ao nome de domínio, você pode excluir o nome de domínio personalizado privado.

Como excluir seu nome de domínio personalizado privado

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, selecione Nomes de domínio personalizados.
3. Escolha seu nome de domínio personalizado privado.
4. Escolha Excluir.
5. Confirme sua escolha e selecione Excluir.

Se necessário, você também pode excluir o endpoint da VPC. Para obter mais informações, consulte [Excluir um endpoint de interface](#).

## AWS CLI

Para limpar

1. O comando `delete-access-association` indicado abaixo exclui a associação de acesso ao nome de domínio:

```
aws apigateway delete-domain-name-access-association \
 --domain-name-access-association-arn 'arn:aws:apigateway:us-
west-2:111122223333:/domainnameaccessassociations/domainname/private.example.com
+abcd1234/vpcsource/vpce-abcd1234efg' \
 --region us-west-2
```

2. O comando `delete-domain-name` indicado abaixo exclui seu nome de domínio personalizado privado. Esse comando também remove todos os mapeamentos de caminho base.

```
aws apigateway delete-domain-name \
 --domain-name test.private.com \
 --domain-name-id abcd1234
```

Se necessário, você também pode excluir o endpoint da VPC. Para obter mais informações, consulte [Excluir um endpoint de interface](#).

## Práticas recomendadas

Sugerimos que você siga as práticas recomendadas indicadas abaixo ao criar um nome de domínio personalizado privado:

- Crie um mapeamento de caminho base ou regras de roteamento para enviar tráfego de um nome de domínio personalizado privado a várias APIs privadas.
- Quando um endpoint da VPC não precisar mais acessar um nome de domínio personalizado privado, exclua a associação. Além disso, remova o endpoint da VPC da `policy` referente ao serviço `execute-api` do domínio personalizado privado.
- Configure pelo menos duas zonas de disponibilidade por endpoint da VPC.
- Desabilite o endpoint padrão. Recomendamos desabilitar o endpoint padrão para permitir que os consumidores de API chamem sua API somente pelo nome de domínio personalizado. Para obter mais informações, consulte [the section called “Desabilitar o endpoint padrão”](#).
- Recomendamos que você provisione uma zona hospedada privada do Route 53 e um registro do tipo A ao configurar seu nome de domínio personalizado privado. Se você não pretende invocar seu nome de domínio personalizado privado, poderá excluir esses recursos posteriormente.

## Como trabalhar com nomes de domínio personalizados privados entre contas

Esta seção explica como trabalhar com nomes de domínio personalizados privados entre contas. É possível fornecer um nome de domínio personalizado privado para outra Conta da AWS e usar outra Conta da AWS para invocar um nome de domínio personalizado privado.

Você pode compartilhar seu nome de domínio personalizado privado com outra Conta da AWS usando o AWS Resource Access Manager ou o API Gateway. O AWS Resource Access Manager (AWS RAM) ajuda você a compartilhar com segurança seus recursos entre Contas da AWS e dentro de sua organização ou unidades organizacionais (UOs). Para obter mais informações, consulte [O que é o AWS Resource Access Manager](#).

Para obter instruções sobre como compartilhar um nome de domínio personalizado privado com outra Conta da AWS usando o AWS RAM, consulte [the section called “Provedor de API: compartilhar seu nome de domínio personalizado privado usando o AWS RAM”](#).

Para obter instruções sobre como compartilhar um nome de domínio personalizado privado com outra Conta da AWS usando o API Gateway, consulte [the section called “Provedor de API: compartilhar seu nome de domínio personalizado privado usando a AWS CLI do API Gateway”](#).

Para obter instruções sobre como consumir um nome de domínio personalizado privado em outra Conta da AWS, consulte [the section called “Consumidor de API: associar o endpoint da VPC a um nome de domínio personalizado privado compartilhado com você”](#).

Práticas recomendadas para trabalhar com nomes de domínio personalizados privados entre contas

Indicamos as seguintes práticas recomendadas para trabalhar com nomes de domínio personalizados privados entre contas:

- Use o AWS RAM para compartilhar nomes de domínio personalizados privados. Ao usar o AWS RAM, você pode reduzir o custo operacional indireto e não precisa criar uma `managementPolicy` para o serviço Amazon API Gateway Management.
- Use o parâmetro `resource-owner` ao listar seus nomes de domínio personalizados privados ou associações de acesso a nomes de domínio. Use o parâmetro `resource-owner` para listar somente os recursos que pertencem a você ou a outras Contas da AWS.

O seguinte exemplo mostra como obter todas as associações de acesso a nomes de domínio pertencentes a você:

```
aws apigateway get-domain-name-access-associations --resource-owner SELF
```

Use `--resource-owner OTHER_ACCOUNTS` para listar todas as associações de acesso a nomes de domínio que outras contas formaram com seu nome de domínio personalizado privado.

Provedor de API: compartilhar seu nome de domínio personalizado privado usando o AWS RAM

Você pode fornecer aos consumidores de API em outras Contas da AWS acesso ao seu nome de domínio personalizado privado. Nesta seção, você aprenderá a compartilhar seu nome de domínio personalizado privado usando o AWS RAM e a controlar o acesso ao seu nome de domínio personalizado privado.

Considerações referentes ao compartilhamento de um nome de domínio personalizado privado

As considerações a seguir podem afetar a forma como você fornece acesso ao seu nome de domínio personalizado privado usando o AWS RAM. Para saber como compartilhar seu nome de domínio personalizado privado sem usar o AWS RAM, consulte [the section called “Provedor de API: compartilhar seu nome de domínio personalizado privado usando a AWS CLI do API Gateway”](#).

- Os nomes de domínio personalizados privados são compartilhados em nível de Região da AWS. Tanto o nome de domínio personalizado privado quanto o endpoint da VPC precisam estar na mesma Região da AWS.
- É possível usar um compartilhamento de recursos com várias entidades principais e, depois de criar o compartilhamento de recursos, adicionar mais entidades principais a ele. Recomendamos que, quando possível, você reutilize seu compartilhamento de recursos.
- Você sempre precisa conceder acesso ao endpoint da VPC do consumidor de API para invocar seu nome de domínio personalizado privado e todas as APIs privadas associadas a ele.
- Se o consumidor e o provedor de API estiverem na mesma organização que está usando o AWS Organizations, o compartilhamento de recursos será aceito automaticamente. Ainda assim você precisa criar o compartilhamento de recursos usando o AWS RAM.
- Se o consumidor e o provedor de API estiverem na mesma organização que está usando o AWS Organizations e o compartilhamento de recursos dentro da organização estiver habilitado, quaisquer entidades principais na organização com as quais você compartilhar receberão acesso automaticamente aos compartilhamentos de recursos. Não há necessidade de convite, e você pode ignorar o compartilhamento de recursos.
- Se o consumidor de API não aceitar o compartilhamento de recursos em 12 horas, o provedor de API deverá compartilhar o recurso novamente.
- Após a criação do compartilhamento de recursos, o AWS RAM atualiza a `managementPolicy` referente ao serviço Amazon API Gateway Management para o nome de domínio personalizado privado a fim de impedir o acesso a entidades principais sem acesso `allow` explícito. Para ter mais informações, consulte [Determining whether a request is allowed or denied within an account](#) no Guia do usuário do IAM.

A managementPolicy atualizada será semelhante ao seguinte:

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"
 }
 }
 }
]
}
```

O AWS RAM impediu que entidades principais sem acesso allow explícito criassem associações de acesso ao seu nome de domínio personalizado privado, adicionando o seguinte:

```
"StringNotEquals": {
 "aws:PrincipalAccount": "111122223333"
}
```

Você ainda pode usar a entidade principal na Conta da AWS que criou o nome de domínio personalizado privado para criar associações de acesso a nomes de domínio.

Permitir que outras contas criem associações de acesso a nomes de domínio com seu nome de domínio personalizado privado

Primeiro, conceda acesso a outra Conta da AWS para criar associações de acesso ao nome de domínio com seu nome de domínio personalizado privado.

### Console de gerenciamento da AWS

Para usar o Console de gerenciamento da AWS, consulte [Creating a resource share in AWS RAM](#) no Guia do usuário do AWS RAM.

Em Selecionar tipo de recurso, escolha Domínios personalizados privados do API Gateway.

### AWS CLI

O [create-resource-share](#) indicado abaixo cria um compartilhamento de recursos para seu nome de domínio personalizado privado. Pode levar alguns minutos para que as associações de entidades principais entre recurso e principal sejam concluídas. Para entidades principais, forneça um ID de conta ou um ID de organização, como `arn:aws:organizations::123456789012:organization/o-1234abcd`. Você pode fornecer várias entidades principais para seu compartilhamento de recursos.

```
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
```

Depois de fornecer acesso a outra Conta da AWS, os consumidores de API dessa conta devem criar uma associação de acesso ao nome de domínio entre o endpoint da VPC deles e seu nome de domínio personalizado privado. Não é possível criar a associação de acesso ao nome de domínio para eles. Para obter mais informações, consulte [the section called “Associate the endpoint of the VPC to a private custom domain name”](#).

Permita que outras contas invoquem seu nome de domínio personalizado privado

Em seguida, conceda acesso ao endpoint da VPC do consumidor de API para invocar seu nome de domínio personalizado privado e todas as APIs privadas associadas a ele.

## Console de gerenciamento da AWS

Como permitir que endpoints da VPC em outras contas invoquem seu nome de domínio personalizado privado

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, selecione Nomes de domínio personalizados.
3. Escolha o nome de domínio personalizado privado que você compartilhou com outras Contas da AWS.
4. Na guia Política de recursos, selecione Editar política de recursos.
5. Adicione o ID do endpoint da VPC do consumidor de API à sua política de recursos.

Você pode encontrar o ID do endpoint da VPC do consumidor de API na seção Associações de acesso a nomes de domínio da guia Compartilhamento de recursos na página Detalhes do domínio do seu nome de domínio personalizado privado.

6. Escolha Salvar alterações.

## AWS CLI

A policy a seguir referente ao serviço `execute-api` permite o tráfego de entrada para um nome de domínio personalizado privado originado dos endpoints da VPC *vpce-abcd1234efg* e *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"
]
 }
 }
}

```

O comando [update-domain-name](#) indicado abaixo usa uma operação de patch para atualizar a policy para um nome de domínio 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
 "]}}}]'

```

Provedor de API: interromper o compartilhamento de um nome de domínio personalizado privado usando o AWS RAM

Para interromper o compartilhamento de seu nome de domínio personalizado privado, primeiro impeça que o consumidor de API crie mais associações de acesso a nomes de domínio dissociando o compartilhamento de recursos. Em seguida, rejeite a associação de acesso ao nome de domínio e remova o endpoint da VPC do consumidor de API da policy referente ao serviço execute-api. O consumidor de API pode então excluir a associação de acesso ao nome de domínio que ele fez.

Interromper o compartilhamento do nome de domínio personalizado privado

Primeiro, interrompa o compartilhamento de recursos usando o AWS RAM.



## Console de gerenciamento da AWS

Para usar o Console de gerenciamento da AWS, consulte [Update a resource share in AWS RAM](#).

## AWS CLI

O comando [disassociate-resource-share](#) indicado abaixo dissocia um compartilhamento de recursos para seu nome de domínio 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
```

## Rejeitar uma associação de acesso ao nome de domínio

Depois de interromper o compartilhamento de seu recurso usando o AWS RAM, rejeite a associação de acesso ao nome de domínio entre um endpoint da VPC em outra conta e seu nome de domínio personalizado privado.

### Note

Não é possível rejeitar uma associação de acesso ao nome de domínio em sua conta. Para interromper o compartilhamento de recursos, exclua a associação de acesso ao nome de domínio. Para ter mais informações, consulte [Excluir uma associação de acesso ao nome de domínio](#).

Quando você rejeita uma associação de acesso a um nome de domínio com um endpoint da VPC, se um consumidor de API tentar chamar seu nome de domínio personalizado privado, o API Gateway rejeitará a chamada e retornará um código de status 403.

## Console de gerenciamento da AWS

Como rejeitar uma associação de acesso ao nome de domínio

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, selecione Nomes de domínio personalizados.

3. Escolha o nome de domínio personalizado privado que você compartilhou com outras Contas da AWS.
4. Em Compartilhamento de recursos, escolha a associação de acesso ao nome de domínio que você deseja rejeitar.
5. Escolha Rejeitar associação.
6. Confirme sua escolha e selecione Rejeitar.

## AWS CLI

O comando `reject-domain-name-access-association` indicado abaixo rejeita a associação de acesso ao nome de domínio entre o endpoint da VPC e seu nome de domínio 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/vpcsource/
vpce-abcd1234efg \
 --domain-name-arn arn:aws:apigateway:us-west-2:111122223333:/domainnames/
private.example.com+abcd1234
```

Negar ao provedor de API acesso para invocar seu nome de domínio personalizado privado

Depois de rejeitar a associação de acesso ao nome de domínio, remova o endpoint da VPC do consumidor de API da sua `policy` referente ao serviço `execute-api`.

## Console de gerenciamento da AWS

Como remover o endpoint da VPC do consumidor de API da sua política de recursos

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, selecione Nomes de domínio personalizados.
3. Escolha o nome de domínio personalizado privado que você compartilhou com outras Contas da AWS.
4. Na guia Política de recursos, escolha Editar.
5. Remova o endpoint da VPC da política.
6. Escolha Salvar alterações.

## AWS CLI

O comando [update-domain-name](#) a seguir usa uma operação de patch para atualizar a policy referente ao serviço `execute-api` de um nome de domínio personalizado privado. Essa nova policy remove um ID de endpoint da VPC adicional acrescentado em [the section called “Permita que outras contas invoquem seu nome de domínio 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"
 }
 }
 }
]
 }'
```

O consumidor de API deve então excluir a associação de acesso ao nome de domínio. Não é possível excluí-lo em nome dele. Para obter mais informações, consulte [the section called “Consumidor de API: excluir a associação de acesso ao nome de domínio com um nome de domínio personalizado privado”](#).

Provedor de API: compartilhar seu nome de domínio personalizado privado usando a AWS CLI do API Gateway

Você pode compartilhar um nome de domínio personalizado privado usando a AWS CLI do API Gateway, mas recomendamos usar o AWS RAM para reduzir o custo operacional indireto. Para obter instruções sobre como usar o AWS RAM para compartilhar nomes de domínio personalizados privados, consulte [the section called “Provedor de API: compartilhar seu nome de domínio personalizado privado usando o AWS RAM”](#).

Para compartilhar um nome de domínio personalizado privado usando a AWS CLI do API Gateway, conceda a outras Contas da AWS acesso para criar associações de acesso a nomes de domínio e invocar seu nome de domínio personalizado privado. Para isso, atualize a `managementPolicy` referente ao serviço API Gateway Management e a policy referente ao serviço `execute-api` do seu nome de domínio personalizado privado. Você também precisa conceder acesso ao endpoint da VPC do consumidor de API na política de recursos referente a todas as APIs privadas associadas ao seu nome de domínio personalizado privado.

O consumidor de API ainda assim precisa criar uma associação de acesso ao nome de domínio em sua própria conta entre o endpoint da VPC dele e seu nome de domínio personalizado privado. Não é possível fazer isso por ele.

## Conceder acesso ao nome de domínio personalizado privado

### Como conceder acesso ao nome de domínio personalizado privado

1. Para atualizar a `managementPolicy` referente ao serviço API Gateway Management, crie um arquivo JSON que contenha a operação de patch para atualizar a política. O comando `patch-managementPolicy.json` a seguir substitui a `managementPolicy` atual por uma política de exemplo que concede às Contas da AWS 111122223333 e 444455556666 acesso para criar associações de acesso a nomes de domínio com o nome de domínio 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\"}]}"
}]
```

O comando [update-domain-name](#) a seguir atualiza a `managementPolicy` usando `patch-managementPolicy.json`.

```
aws apigateway update-domain-name \
 --domain-name private.example.com \
 --domain-name-id abcd1234 \
 --patch-operations file://patch-managementPolicy.json
```

Depois de conceder acesso, você precisa notificar o consumidor de API de que ele pode formar a associação de acesso ao nome de domínio. Se você usar o AWS RAM, o AWS RAM executará essa etapa para você.

2. Para atualizar a `policy` referente ao serviço `execute-api`, crie um arquivo JSON que contenha a operação de patch para atualizar a política. O exemplo `patch-policy.json` a seguir substitui a `policy` atual por um exemplo de política que concede acesso

a dois endpoints da VPC para invocar o nome de domínio 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\"]}}}]}"
}]
```

Use o comando [update-domain-name](#) a seguir para atualizar a policy usando `patch-policy.json`.

```
aws apigateway update-domain-name \
 --domain-name private.example.com \
 --domain-name-id abcd1234 \
 --patch-operations file://patch-policy.json
```

## Negar acesso ao nome de domínio personalizado privado

Para interromper o compartilhamento de seu nome de domínio personalizado privado, você precisa rejeitar a associação de acesso ao nome de domínio entre seu nome de domínio personalizado privado e o endpoint da VPC do consumidor de API.

### Como negar acesso ao nome de domínio personalizado privado

1. O comando `reject-domain-name-access-association` a seguir rejeita a associação de acesso ao nome de domínio.

```
aws apigateway reject-domain-name-access-association \
 --domain-name-access-association-arn arn:aws:apigateway:us-west-2:444455556666:/domainnameaccessassociations/domainname/private.example.com+abcd1234/vpcsource/vpce-abcd1234 \
 --domain-name-arn arn:aws:apigateway:us-west-2:111122223333:/domainnames/private.example.com+abcd1234
```

2. Modifique o `patch-managementPolicy.json` para remover o acesso à conta do provedor de API e criar uma associação de acesso ao nome de domínio com seu nome de domínio personalizado privado. O seguinte `patch-managementPolicy.json` remove uma conta da `managementPolicy`:

```
[{
 "op": "replace",
 "path": "/managementPolicy",
 "value": "{\"Version\":\"2012-10-17\", \"Statement\": [{\"Effect\": \"Allow\", \"Principal\": \"*\", \"Action\": \"apigateway:CreateAccessAssociation\", \"Resource\": \"arn:aws:apigateway:us-west-2:11112223333:/domainnames/private.example.com+abcd1234\"}]}"
}]
```

O comando [update-domain-name](#) a seguir atualiza a `managementPolicy` usando `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 o `patch-policy.json` para remover o acesso do endpoint da VPC do provedor de API para invocar seu nome de domínio personalizado privado. O seguinte `patch-policy.json` remove o ID do endpoint da VPC da `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-2:11112223333:/domainnames/private.example.com+abcd1234\"}, {\"Effect\": \"Deny\", \"Principal\": \"*\", \"Action\": \"execute-api:Invoke\", \"Resource\": \"arn:aws:execute-api:us-west-2:11112223333:/domainnames/private.example.com+abcd1234\", \"Condition\": {\"StringNotEquals\": {\"aws:SourceVpce\": \"vpce-abcd1234\"}}}]}"
}]
```

O comando [update-domain-name](#) a seguir atualiza a `policy` usando `patch-policy.json`.

```
aws apigateway update-domain-name \
 --domain-name private.example.com \
 --domain-name-id abcd1234 \
 --patch-operations file://patch-policy.json
```

## Exemplos de políticas usadas neste procedimento

A seção a seguir mostra exemplos de políticas usadas no procedimento anterior.

O exemplo de política a seguir refere-se à `managementPolicy` do serviço Amazon API Gateway Management. Essa política concede acesso às Contas da AWS 111122223333 e 444455556666 para criar associações de acesso a nomes de domínio com o nome de domínio 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"
 }
]
}
```

O exemplo de política a seguir refere-se à `policy` do serviço `execute-api`. Essa política concede aos endpoints da VPC `vpce-abcd1234` e `vpce-xyzz0000` acesso para invocar o nome de domínio 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: associar o endpoint da VPC a um nome de domínio personalizado privado compartilhado com você

O procedimento a seguir mostra como consumir um nome de domínio privado em outra Conta da AWS. Dependendo da sua relação de confiança com o provedor de API, o AWS RAM pode concluir algumas tarefas para você.

Quando você está em uma Conta da AWS diferente da conta de um nome de domínio personalizado privado, só é possível associar o endpoint da VPC a um nome de domínio personalizado privado e invocá-lo. Não é possível visualizar a `policy` nem nenhum outro parâmetro do nome de domínio personalizado privado.



## Pré-requisitos

Para consumir um nome de domínio personalizado privado em outra Conta da AWS, é necessário observar os seguintes pré-requisitos:

- Uma VPC e um endpoint da VPC para o serviço `execute-api`. A VPC deve ter `enableDnsHostnames` e `enableDnsSupport` configurados como `true`.
- Recomendamos configurar pelo menos duas zonas de disponibilidade por endpoint da VPC.

(Opcional) Aceite o compartilhamento de recursos do domínio personalizado privado

Se o provedor de API usou o AWS RAM para criar um compartilhamento de recursos, você tem 12 horas para aceitá-lo. Se você estiver na mesma organização que está usando o AWS Organizations como provedor de API, o compartilhamento será aceito automaticamente. Se estiver em uma organização que tem recursos compartilhados automáticos habilitados, o recurso será compartilhado automaticamente com você.

## Console de gerenciamento da AWS

Para usar o Console de gerenciamento da AWS, consulte [Accepting and rejecting resource share invitations](#) no Guia do usuário do AWS RAM.

## AWS CLI

Para encontrar todos os recursos compartilhados com você, use o comando [get-resource-share-invitations](#) indicado abaixo:

```
aws ram get-resource-share-invitations \
 --region us-west-2
```

Use o ARN de compartilhamento de recursos resultante para aceitar o convite de compartilhamento de recursos. O comando [accept-resource-share-invitation](#) a seguir aceita o compartilhamento de recursos.

```
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
```

## Associe o endpoint da VPC a um nome de domínio personalizado privado compartilhado

Como os nomes de domínio personalizados privados não são exclusivos, você deve associar o endpoint da VPC ao ARN do nome de domínio personalizado exclusivo. Após a criação da associação de acesso ao nome de domínio, pode demorar até 15 minutos para que o endpoint da VPC invoque com sucesso o nome de domínio personalizado privado. Se você usa um endpoint da VPC para acessar um nome de domínio personalizado público, não o use para criar associações de acesso a nomes de domínio.

### Console de gerenciamento da AWS

Como associar o endpoint da VPC a um nome de domínio personalizado privado compartilhado

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, escolha Associações de acesso a nomes de domínio.
3. Escolha Criar uma associação de acesso ao nome de domínio.
4. Em ARN do nome de domínio, selecione o ARN do nome de domínio que o provedor de API compartilhou com você.

O ARN do nome de domínio pode não aparecer na lista suspensa. Você pode usar o console do AWS RAM para visualizar os nomes de domínio compartilhados com você e, em seguida, copiar o ARN do nome de domínio e inseri-lo nesse campo.

5. Em IDs de endpoint da VPC, selecione o ID do endpoint da VPC com o qual você deseja formar a associação de acesso ao nome de domínio.
6. Escolha Criar uma associação de acesso ao nome de domínio.

### AWS CLI

Como os nomes de domínio personalizados privados não são exclusivos, você deve associar o endpoint da VPC ao ARN do nome de domínio personalizado exclusivo. Para localizar o ARN do nome de domínio, use um dos comandos a seguir.

1. AWS RAM

Você pode usar o comando [list-resources](#) a seguir para visualizar os recursos que são compartilhados com você. Para usar esse comando, o provedor de API deve ter usado o AWS RAM para compartilhar o respectivo domínio personalizado privado com você.

```
aws ram list-resources \
 --resource-owner OTHER-ACCOUNTS \
 --region us-west-2
 --resource-type apigateway:Domainnames
```

## API Gateway

O comando `get-domain-names` a seguir lista todos os nomes de domínio personalizados privados pertencentes a outras Contas da AWS com as quais você pode formar associações de acesso a nomes de domínio.

```
aws apigateway get-domain-names \
 --resource-owner OTHER_ACCOUNTS \
 --region us-west-2
```

2. Depois de recuperar o ARN, use o API Gateway para criar a associação de acesso ao nome de domínio entre o endpoint da VPC e um nome de domínio personalizado privado compartilhado. Use o seguinte 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"
```

A saída será exibida da seguinte forma:

```
{
 "domainNameAccessAssociationARN": "arn:aws:apigateway:us-
west-2:444455556666:/domainnameaccessassociations/domainname/private.example.com
+abcd1234/vpcsource/vpce-abcd1234efg",
 "accessAssociationSource": "vpce-1a2b3c4d5e6f1a2b3",
 "accessAssociationSourceType": "VPCE",
 "domainNameARN" : "arn:aws:apigateway:us-west-1:111122223333:/domainnames/
private.example.com+a1b2c3"
}
```

Depois de associar o endpoint da VPC ao nome de domínio personalizado privado, confirme se o provedor de API atualizou a política do nome de domínio personalizado privado para permitir que

seu endpoint da VPC invoque o nome de domínio. Para obter mais informações, consulte [the section called “Permita que outras contas invoquem seu nome de domínio personalizado privado”](#).

## Como criar uma zona hospedada no Route 53

Para resolver o nome de domínio personalizado privado, você precisa criar uma zona hospedada privada do Route 53. Zona hospedada é um contêiner que abriga informações sobre como você deseja rotear o tráfego para um domínio dentro de uma ou mais VPCs sem expor seus recursos à internet. Para obter mais informações, consulte [Como trabalhar com zonas hospedadas privadas](#).

### Console de gerenciamento da AWS

Para usar o Console de gerenciamento da AWS, consulte [Creating a private hosted zone](#) no Guia do desenvolvedor do Amazon Route 53.

Em Name, use o nome do domínio personalizado privado. Em ID da VPC, use a VPC que contém o endpoint da VPC que você usou para a associação de acesso ao nome de domínio.

### AWS CLI

O comando [create-hosted-zone](#) indicado abaixo cria uma zona hospedada 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
```

A saída contém o ID da zona hospedada. Você usa o ID da zona hospedada nas etapas a seguir.

### Crie um registro de DNS A do Route 53.

Depois de criar a zona hospedada, crie um registro para resolver o domínio personalizado privado. Neste exemplo, você cria um tipo de registro A. Se estiver usando IPv6 para o endpoint da VPC, crie um tipo de registro AAAA. Se estiver usando pilha dupla para o endpoint da VPC, crie um registro do tipo AAAA e um do tipo A.

### Console de gerenciamento da AWS

Para usar o Console de gerenciamento da AWS, consulte [Routing traffic to an Amazon API Gateway API by using your domain name](#).

Use Criação rápida e ative Alias. Para o endpoint, use o nome DNS do endpoint da VPC.

## AWS CLI

Para configurar os registros de DNS para associar o nome de domínio personalizado privado ao nome de host do ID de zona hospedada fornecido, primeiro crie um arquivo JSON que contenha a configuração de um registro de DNS para o nome de domínio privado.

O comando `setup-dns-record.json` a seguir mostra como criar um registro de DNS A para associar um nome de domínio personalizado privado ao respectivo nome de host privado. Você fornece o `DNSName` do ID do DNS da VPC e o ID da zona hospedada que você criou na etapa anterior.

```
{
 "Changes": [
 {
 "Action": "UPSERT",
 "ResourceRecordSet": {
 "Name": "private.example.com",
 "Type": "A",
 "AliasTarget": {
 "DNSName": "vpce-abcd1234.execute-api.us-west-2.vpce.amazonaws.com",
 "HostedZoneId": "Z20JLYMU09EFXC",
 "EvaluateTargetHealth": false
 }
 }
 }
]
}
```

O comando [change-resource-record-sets](#) indicado abaixo cria um registro de DNS para o nome de domínio personalizado privado:

```
aws route53 change-resource-record-sets \
 --hosted-zone-id ZABCEFG1234 \
 --change-batch file:///path/to/your/setup-dns-record.json
```

Substitua `hosted-zone-id` pelo ID da zona hospedada do Route 53 do conjunto de registros DNS na conta. O valor do parâmetro `change-batch` aponta para um arquivo JSON.

## Próximas etapas para um consumidor de API

Agora você pode invocar a API privada em sua Conta da AWS. Na VPC, você pode usar o comando curl a seguir para acessar o nome de domínio personalizado privado.

```
curl https://private.example.com/v1
```

Para ter mais informações sobre outras formas de invocar sua API privada, consulte [the section called “Invocar uma API privada usando um nome de domínio personalizado”](#).

Consumidor de API: excluir a associação de acesso ao nome de domínio com um nome de domínio personalizado privado

Se você for um consumidor de API, poderá excluir o recurso de associação de acesso a qualquer momento. O provedor de API não pode excluir a associação de acesso ao nome de domínio para você.

Recomendamos que você sempre exclua uma associação de acesso ao nome de domínio quando não a estiver mais usando.

## Console de gerenciamento da AWS

Como excluir a associação de acesso ao nome de domínio

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, escolha Associações de acesso a nomes de domínio.
3. Selecione sua associação de acesso ao nome de domínio e escolha Excluir.
4. Confirme sua escolha e selecione Excluir.

## AWS CLI

O comando `delete-access-association` indicado abaixo exclui a associação de acesso:

```
aws apigateway delete-domain-name-access-association \
 --domain-name-access-association-arn 'arn:aws:apigateway:us-
west-2:444455556666:/domainnameaccessassociations/domainname/private.example.com
+abcd1234/vpcsource/vpce-abcd1234efg'
```

## Criar um nome de domínio personalizado para APIs privadas usando o CloudFormation

O modelo de exemplo do CloudFormation a seguir cria uma API privada e um nome de domínio personalizado privado, associa a API privada ao nome de domínio personalizado e, em seguida, cria uma associação de acesso ao nome de domínio. Você precisa fornecer seu endpoint da VPC, nome de domínio e ARN do certificado.

As seguintes considerações podem afetar o uso do CloudFormation para criar um nome de domínio personalizado privado:

- Não é possível rejeitar uma associação de acesso a um nome de domínio usando o CloudFormation. Para rejeitar uma associação de acesso a um nome de domínio, use o a AWS CLI.
- Use a propriedade `AWS::ApiGateway::DomainNameV2` do CloudFormation para criar um nome de domínio personalizado privado.
- Use a propriedade `AWS::ApiGateway::BasePathMappingV2` do CloudFormation para criar um mapeamento de caminho 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
```

## Invocar uma API privada

Só é possível invocar uma API privada de dentro de uma VPC usando um endpoint da VPC. A API privada deve ter uma política de recursos que permita que VPCs e endpoints da VPC específicos invoquem a API.

Se você invocar uma API privada sem utilizar um nome de domínio personalizado ou nomes DNS privados e suas APIs ou o nome de domínio usar uma política de segurança que comece com `SecurityPolicy_`, você deverá definir o modo de acesso ao endpoint como BASIC. Para obter mais informações, consulte [the section called “Modo de acesso ao endpoint”](#).

### Invocar uma API privada usando um nome de domínio personalizado

Para invocar uma API privada usando um nome de domínio personalizado, o endpoint da VPC precisa de uma associação de acesso ao nome de domínio com um nome de domínio personalizado e o nome de domínio personalizado precisa permitir o acesso do endpoint da VPC para invocá-lo. Para obter mais informações, consulte [the section called “Nomes de domínio personalizados para APIs privadas”](#).

Não há diferença entre invocar um nome de domínio personalizado privado em uma VPC em sua própria Conta da AWS ou em uma Conta da AWS diferente.

### Use seu nome de domínio personalizado

Dentro da sua VPC, você pode invocar sua API usando o nome de domínio personalizado. O seguinte exemplo mostra um comando curl para invocar seu nome de domínio personalizado privado:

```
curl https://private.example.com
```

### Usar nomes de host DNS privados específicos do endpoint

É possível invocar a API usando o nome de domínio personalizado e o nome de host DNS privado específico do endpoint.

```
curl https://private-dns-hostname.execute-api.region.vpce.amazonaws.com/basepath -H
'Host:custom-domain-name'
```

O exemplo a seguir mostra um comando curl para invocar o nome de domínio personalizado usando um nome de host DNS privado específico do endpoint:

```
curl https://vpce-123456-abc000.execute-api.us-east-2.vpce.amazonaws.com/test -H
'Host:private.example.com'
```

## Invocar uma API privada sem usar um nome de domínio personalizado

Para invocar a API privada sem usar um nome de domínio personalizado, é necessário identificar os nomes DNS da API. O procedimento a seguir mostra como encontrar os nomes DNS.

### Console de gerenciamento da AWS

#### Como encontrar os nomes DNS

1. Faça login no Console de gerenciamento da AWS e abra o console da Amazon VPC em <https://console.aws.amazon.com/vpc/>.
2. No painel de navegação principal, selecione Endpoints e escolha o endpoint de interface da VPC do API Gateway.
3. No painel Detalhes, você verá cinco valores no campo Nomes DNS. Os três primeiros são os nomes DNS públicos da API. Os outros dois são os nomes DNS privados dela.

### AWS CLI

Use o comando [describe-vpc-endpoints](#) a seguir para listar os valores de DNS.

```
aws ec2 describe-vpc-endpoints --vpc-endpoint-ids vpce-01234567abcdef012
```

Os três primeiros são os nomes DNS públicos da API. Os outros dois são os nomes DNS privados dela.

## Invocar uma API privada usando um alias Route53

É possível associar ou desassociar um endpoint da VPC de uma API privada. Para obter mais informações, consulte [the section called “\(Optional\) Associar ou desassociar um endpoint da VPC a uma API privada”](#).

Depois de associar endpoints da VPC à API privada, você pode usar o seguinte URL base para invocar a API:

```
https://{rest-api-id}-{vpce-id}.execute-api.{region}.amazonaws.com/{stage}
```

Por exemplo, se você configurou o método GET /pets para o estágio test e seu ID de API REST foi 01234567ab, seu ID de endpoint da VPC foi vpce-01234567abcdef012 e sua região foi us-west-2, poderá invocar a API como:

```
curl -v https://01234567ab-vpce-01234567abcdef012.execute-api.us-west-2.amazonaws.com/test/pets
```

Invocar uma API privada usando nomes DNS privados

Se você tiver habilitado a opção de DNS privado, poderá acessar a API privada usando o seguinte nome DNS privado:

```
{restapi-id}.execute-api.{region}.amazonaws.com
```

A URL de base para invocar a API está neste formato:

```
https://{restapi-id}.execute-api.{region}.amazonaws.com/{stage}
```

Por exemplo, se você configurou o método GET /pets para o estágio test e seu ID de API REST foi 01234567ab e sua região foi us-west-2, pode invocar a API privada inserindo o seguinte URL em um navegador:

```
https://01234567ab.execute-api.us-west-2.amazonaws.com/test/pets
```

Também é possível usar o seguinte comando cURL para invocar a API privada:

```
curl -X GET https://01234567ab.execute-api.us-west-2.amazonaws.com/test/pets
```

#### Warning

Se habilitar a opção de DNS privado para o endpoint da VPC, você não poderá acessar o endpoint padrão para APIs públicas. Para obter mais informações, consulte [Por que não consigo me conectar à API pública em um endpoint da VPC do API Gateway?](#).

## Invocar uma API privada usando o Direct Connect

Você pode usar o Direct Connect para estabelecer uma conexão privada dedicada de uma rede on-premises com o Amazon VPC e acessar o endpoint da API privada nessa conexão usando nomes DNS públicos.

Você também pode usar nomes de um DNS privado para acessar a sua API privada em uma rede on-premises configurando um endpoint de entrada do Amazon Route 53 Resolver e encaminhando para ele todas as consultas de DNS do DNS privado pela sua rede remota. Para obter mais informações, consulte [Encaminhamento de consultas de DNS de entrada para as suas VPCs](#) no Guia do desenvolvedor do Amazon Route 53.

### Invocar uma API privada usando nomes de host DNS públicos específicos do endpoint

É possível acessar a API privada usando nomes de host DNS específicos de endpoint. Esses são os nomes de hosts DNS públicos que contêm o ID do VPC endpoint ou ID de API da API privada.

O URL base gerado está no seguinte formato:

```
https://{public-dns-hostname}.execute-api.{region}.vpce.amazonaws.com/{stage}
```

Por exemplo, se você configurar o método GET /pets para o estágio test e o ID de API REST for abc1234, o nome de host DNS público for vpce-def-01234567 e a região for us-west-2, você poderá invocar a API privada usando o ID de VPCe dela com o cabeçalho Host em um 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'
```

Também é possível invocar a API privada pelo ID de API usando o cabeçalho x-apigw-api-id em um comando cURL no seguinte formato:

```
curl -v https://{public-dns-hostname}.execute-api.{region}.vpce.amazonaws.com/{stage} -H 'x-apigw-api-id:{api-id}'
```

## Monitorar APIs REST no API Gateway

Nesta seção, você aprenderá a monitorar sua API usando métricas do CloudWatch, CloudWatch Logs, Firehose e AWS X-Ray. Combinando os logs de execução e as métricas do CloudWatch, você pode registrar erros e rastreamentos de execução em log e monitorar a performance da API. Você

também pode registrar em log chamadas de API no Firehose. Você também pode usar o AWS X-Ray para rastrear chamadas por meio dos serviços downstream que compõem a API.

#### Note

O API Gateway pode não gerar logs e métricas nos seguintes casos:

- Erros 413 de entidade de solicitação muito grande
- Erros 431 de campos de cabeçalho de solicitação muito grandes
- Erros 429 de muitas solicitações excessivos
- Erros da série 400 de solicitações enviadas a um domínio personalizado que não tem mapeamento de API
- Erros da série 500 causados por falhas internas

O API Gateway não gerará logs e métricas ao testar um método de API REST. As entradas do CloudWatch são simuladas. Para obter mais informações, consulte [the section called “Usar o console para testar um método de API REST”](#).

## Tópicos

- [Monitorar a execução da API REST com métricas do Amazon CloudWatch](#)
- [Configurar o registro em log do CloudWatch para APIs REST no API Gateway](#)
- [Registrar em log chamadas de API REST para o Amazon Data Firehose no API Gateway](#)
- [Variáveis para registro em log de acesso no API Gateway](#)
- [Rastrear solicitações de usuário para APIs REST usando o X-Ray no API Gateway](#)

## Monitorar a execução da API REST com métricas do Amazon CloudWatch

É possível monitorar a execução da API usando o CloudWatch, que coleta e processa dados brutos do API Gateway em métricas legíveis, quase em tempo real. Essas estatísticas são registradas para um período de 15 meses, de forma que você possa acessar informações históricas e ganhar uma perspectiva melhor sobre como seu serviço ou aplicativo web está se saindo. Por padrão, os dados de métricas do API Gateway são enviados automaticamente para o CloudWatch em períodos de um minuto. Para obter mais informações, consulte [O que é o Amazon CloudWatch?](#) no Guia do usuário do Amazon CloudWatch.

As métricas informadas pelo API Gateway fornecem informações que podem ser analisadas de diferentes maneiras. A lista a seguir mostra alguns usos comuns de métricas que são sugestões para começar a usar:

- Monitore as métricas de IntegrationLatency para medir a capacidade de resposta do backend.
- Monitore as métricas de Latency para medir a capacidade de resposta geral das suas chamadas de API.
- Monitore as métricas de CacheHitCount e CacheMissCount para otimizar capacidades de cache de modo a alcançar o desempenho desejado.

## Tópicos

- [Dimensões e métricas do Amazon API Gateway](#)
- [Exibir métricas do CloudWatch com o painel de API no API Gateway](#)
- [Exibir métricas do API Gateway no console do CloudWatch](#)
- [Exibir eventos de log do API Gateway no console do CloudWatch](#)
- [Ferramentas de monitoramento na AWS para o API Gateway](#)

## Dimensões e métricas do Amazon API Gateway

As métricas e dimensões que o API Gateway envia para o Amazon CloudWatch estão listadas abaixo. Para obter mais informações, consulte [Monitorar a execução da API REST com métricas do Amazon CloudWatch](#).

### Métricas do API Gateway

O Amazon API Gateway envia dados de métricas para o CloudWatch a cada minuto.

O namespace AWS/ApiGateway inclui as métricas a seguir.

| Métrica  | Descrição                                                                                                                                                                                     |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4XXError | <p>O número de erros no lado do cliente capturados em um determinado período.</p> <p>O API Gateway considera os códigos de status de resposta do gateway modificados como erros 4XXError.</p> |

| Métrica       | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|               | <p>A estatística Sum representa essa métrica, ou seja, a contagem total de erros 4XXError no período especificado. A estatística Average representa a taxa de erros 4XXError, ou seja, a contagem total de erros 4XXError dividida pelo número total de solicitações durante o período. O denominador corresponde à métrica Count (abaixo).</p> <p>Unit: Count</p>                                                                                              |
| 5XXError      | <p>O número de erros do servidor capturados em um período determinado.</p> <p>A estatística Sum representa essa métrica, ou seja, a contagem total de erros 5XXError no período especificado. A estatística Average representa a taxa de erros 5XXError, ou seja, a contagem total de erros 5XXError dividida pelo número total de solicitações durante o período. O denominador corresponde à métrica Count (abaixo).</p> <p>Unit: Count</p>                   |
| CacheHitCount | <p>O número de solicitações atendidas pelo cache da API em um determinado período.</p> <p>A estatística Sum representa essa métrica, ou seja, a contagem total de acertos de cache no período especificado. A estatística Average representa a taxa de acertos de cache, a saber, a contagem total de acertos de cache dividida pelo número total de solicitações durante o período. O denominador corresponde à métrica Count (abaixo).</p> <p>Unit: Count</p> |



| Métrica            | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CacheMissCount     | <p>O número de solicitações atendidas pelo backend em um determinado período quando o armazenamento em cache da API está habilitado.</p> <p>A estatística Sum representa essa métrica, ou seja, a contagem total de erros de cache no período especificado. A estatística Average representa a taxa de erros de cache, ou seja, a contagem total de erros de cache dividida pelo número total de solicitações durante o período. O denominador corresponde à métrica Count (abaixo).</p> <p>Unit: Count</p> |
| Count              | <p>O número total de solicitações de API em um determinado período.</p> <p>A estatística SampleCount representa essa métrica.</p> <p>Unit: Count</p>                                                                                                                                                                                                                                                                                                                                                        |
| IntegrationLatency | <p>O tempo que o API Gateway leva para receber uma resposta do backend depois de retransmitir uma solicitação para o backend.</p> <p>Unit: Millisecond</p>                                                                                                                                                                                                                                                                                                                                                  |
| Latency            | <p>O tempo que o API Gateway leva para devolver uma resposta para um cliente depois de receber uma solicitação do cliente. A latência inclui a latência de integração e outras despesas gerais do API Gateway.</p> <p>Unit: Millisecond</p>                                                                                                                                                                                                                                                                 |

## Dimensões para métricas

É possível usar as dimensões na tabela a seguir para filtrar métricas do API Gateway.

### Note

O API Gateway remove caracteres não ASCII da dimensão ApiName antes de enviar métricas para o CloudWatch. Se o ApiName contiver caracteres ASCII, o API ID será usado como o ApiName.

| Dimensão                         | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ApiName                          | Filtra as métricas do API Gateway para a API REST com o nome de API especificado.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| ApiName, Method, Resource, Stage | <p>Filtra as métricas do API Gateway para o método de API com o nome, o estágio, o recurso e o método de API especificados.</p> <p>O API Gateway não enviará essas métricas a menos que você tenha habilitado explicitamente métricas detalhadas do CloudWatch. No console, selecione um estágio e, em Logs e rastreamento, selecione Editar. Selecione Métricas detalhadas e, depois, Salvar alterações. Como alternativa, você pode chamar o comando <a href="#">update-stage</a> da AWS CLI para atualizar a propriedade <code>metricsEnabled</code> para <code>true</code>.</p> <p>Habilitar essas métricas incorrerá em cobranças adicionais na conta. Para obter informações de definição de preço, consulte <a href="#">Definição de preço do Amazon CloudWatch</a>.</p> |
| ApiName, Stage                   | Filtra as métricas do API Gateway para o recurso de estágio de API com o nome e o estágio de API especificados.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

## Exibir métricas do CloudWatch com o painel de API no API Gateway

Você pode usar o painel de API no console do API Gateway para exibir as métricas do CloudWatch da API implantada no API Gateway. Elas são apresentadas como um resumo da atividade da API ao longo do tempo.

### Tópicos

- [Pré-requisitos](#)
- [Examinar atividades da API no painel](#)

### Pré-requisitos

1. Você deve ter uma API criada no API Gateway. Siga as instruções em [Desenvolver APIs REST no API Gateway](#).
2. Você deve ter a API implantada pelo menos uma vez. Siga as instruções em [Implantar APIs REST no API Gateway](#).

### Examinar atividades da API no painel

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API.
3. No painel de navegação principal, escolha Painel.
4. Em Estágio, escolha o estágio desejado.
5. Escolha Intervalo de datas para especificar um intervalo de datas.
6. Se necessário, atualize e visualize as métricas individuais exibidas em grafos separados, chamados Chamadas de API, Latência, Latência de integração, Latência, Erro 4xx e Erro 5xx.

#### Tip

Para examinar métricas do CloudWatch em nível de método, verifique se o CloudWatch Logs está habilitado em nível de método. Para obter mais informações sobre como configurar o registro em log em nível de método, consulte [Substituir configurações em nível de estágio](#).

## Exibir métricas do API Gateway no console do CloudWatch

As métricas são agrupadas primeiro pelo namespace do serviço e, em seguida, por várias combinações de dimensão dentro de cada namespace. Para visualizar as métricas em nível de método da API, ative as métricas detalhadas. Para obter mais informações, consulte [the section called “Atualizar configurações de estágio”](#).

Como exibir métricas do API Gateway usando o console do CloudWatch

1. Abra o console do CloudWatch, em <https://console.aws.amazon.com/cloudwatch/>.
2. Se necessário, altere a Região da AWS. Na barra de navegação, selecione a região em que os seus recursos da AWS residem.
3. No painel de navegação, selecione Métricas.
4. Na guia All metrics (Todas as métricas), escolha API Gateway.
5. Para visualizar as métricas por estágio, escolha o painel By Stage (Por estágio). E selecione as APIs e os nomes de métrica.
6. Para visualizar as métricas por API específica, escolha o painel By Api Name (Por nome de Api). E selecione as APIs e os nomes de métrica.

Para visualizar métricas usando a CLI da AWS

1. Use o comando [list-metrics](#) indicado abaixo para listar as métricas:

```
aws cloudwatch list-metrics --namespace "AWS/ApiGateway"
```

Depois de criar uma métrica, aguarde até 15 minutos para que ela apareça. Para ver as estatísticas métricas com antecedência, use [get-metric-data](#) ou [get-metric-statistics](#).

2. Use o comando [get-metrics-statistics](#) indicado abaixo para visualizar a média ao longo de um período usando 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
```

## Exibir eventos de log do API Gateway no console do CloudWatch

A seção a seguir explica os pré-requisitos e como visualizar os eventos de log do API Gateway no console do CloudWatch.

### Pré-requisitos

1. Você deve ter uma API criada no API Gateway. Siga as instruções em [Desenvolver APIs REST no API Gateway](#).
2. Você deve ter implantado e invocado a API pelo menos uma vez. Siga as instruções em [Implantar APIs REST no API Gateway](#) e [Invocar APIs REST no API Gateway](#).
3. Você deve ter o CloudWatch Logs habilitado para um estágio. Siga as instruções em [Configurar o registro em log do CloudWatch para APIs REST no API Gateway](#).

Como exibir solicitações e respostas de APIs registradas em log usando o console do CloudWatch

1. Abra o console do CloudWatch, em <https://console.aws.amazon.com/cloudwatch/>.
2. Se necessário, altere a Região da AWS. Na barra de navegação, selecione a região em que os seus recursos da AWS residem. Para obter mais informações, consulte [Regiões e endpoints](#).
3. No painel de navegação, selecione Logs, Grupos de log.
4. Na tabela Log Groups (Grupos de logs), selecione um grupo de logs do nome API-Gateway-Execution-Logs\_{rest-api-id}/{stage-name}.
5. Na tabela Log Streams (Fluxos de log), escolha um fluxo de logs. Você pode usar o carimbo de data e hora para ajudar a localizar o fluxo de logs de seu interesse.
6. Escolha Text (Texto) para visualizar texto bruto ou escolha Row (Linha) para visualizar o evento linha por linha.

### Important

O CloudWatch permite que você exclua streams ou grupos de logs. Não exclua manualmente os streams ou grupos de logs da API do API Gateway. Deixe o API Gateway gerenciar esses recursos. A exclusão manual de grupos ou fluxos de log pode fazer com que as solicitações e as respostas da API não sejam registradas. Se isso acontecer, você pode excluir todo o grupo de logs da API e reimplantá-la. Isso ocorre porque o API Gateway cria grupos ou fluxos de log para um estágio de API no momento em que ela é implantada.

## Ferramentas de monitoramento na AWS para o API Gateway

AWSA fornece várias ferramentas que podem ser usadas para monitorar o API Gateway. É possível configurar algumas dessas ferramentas para realizar o monitoramento automaticamente, enquanto outras exigem intervenção manual. Recomendamos que as tarefas de monitoramento sejam automatizadas ao máximo possível.

### Ferramentas de monitoramento automatizadas na AWS

É possível usar as seguintes ferramentas de monitoramento automatizadas para supervisionar o API Gateway e gerar relatórios quando algo estiver errado:

- Amazon CloudWatch Alarms: observe uma única métrica ao longo de um período que você especificar e realize uma ou mais ações com base no valor da métrica em relação a um limite ao longo de vários períodos. A ação é uma notificação enviada para um tópico do Amazon Simple Notification Service (Amazon SNS) ou uma política do Amazon EC2 Auto Scaling. Os alarmes do CloudWatch não invocam ações simplesmente por estarem em um estado específico. O estado deve ter sido alterado e mantido por um número específico de períodos. Para obter mais informações, consulte [Monitorar a execução da API REST com métricas do Amazon CloudWatch](#).
- Amazon CloudWatch Logs: monitore, armazene e acesse seus arquivos de log do AWS CloudTrail ou de outras origens. Para obter mais informações, consulte [What is CloudWatch Logs?](#) no Guia do usuário do Amazon CloudWatch.
- Amazon EventBridge (anteriormente chamado de CloudWatch Events): faça correspondência de eventos e direcione-os a uma ou mais funções ou streams de destino para fazer alterações, capturar informações de estado e realizar ações corretivas. Para obter mais informações, consulte [What Is Amazon EventBridge?](#) no Guia do usuário do Amazon EventBridge.
- Monitoramento de log AWS CloudTrail: compartilhe arquivos de log entre contas, monitore os arquivos de log do CloudTrail em tempo real enviando-os para o CloudWatch Logs, escreva aplicações de processamento de logs em Java e confirme se os arquivos de log não foram alterados após a entrega pelo CloudTrail. Para obter mais informações, consulte [Trabalhando com arquivos de log do CloudTrail](#) no Guia do usuário do AWS CloudTrail.

### Ferramentas de monitoramento manual

Outra parte importante do monitoramento do API Gateway é o monitoramento manual dos itens que os alarmes do CloudWatch não abrangem. O API Gateway, o CloudWatch e outros painéis do console da AWS fornecem uma visualização rápida do estado de seu ambiente da AWS. Recomendamos que você também verifique os arquivos de log na execução da API.

- O painel do API Gateway mostra as seguintes estatísticas para um estágio de API durante um período específico:
  - API Calls (Chamadas de API)
  - Cache Hit (Acertos do cache), apenas quando o armazenamento em cache de API está ativado.
  - Cache Miss (Solicitações não atendidas pelo cache), apenas quando o armazenamento em cache de API está ativado.
  - Latência
  - Integration Latency (Latência de integração)
  - 4XX Error (Erro 4XX)
  - 5XX Error (Erro 5XX)
- A página inicial do CloudWatch mostra:
  - Alertas e status atual
  - Gráficos de alertas e recursos
  - Estado de integridade do serviço

Além disso, é possível usar o CloudWatch para fazer o seguinte:

- Crie [painéis personalizados](#) para monitorar os serviços com os quais você se preocupa.
- Colocar em gráfico dados de métrica para solucionar problemas e descobrir tendências
- Pesquisar e procurar todas as métricas de recursos da AWS
- Criar e editar alertas para ser notificado sobre problemas

## Criar alarmes do CloudWatch para monitorar o API Gateway

Você pode criar um alarme do CloudWatch que envia uma mensagem do Amazon SNS quando o alarme muda de estado. Um alarme observa uma única métrica ao longo de um período especificado por você e realiza uma ou mais ações com base no valor da métrica relativo a um determinado limite ao longo de vários períodos. A ação é uma notificação enviada para um tópico do Amazon SNS ou uma política de Auto Scaling. Os alertas invocam ações apenas para alterações de estado mantidas. Os alarmes do CloudWatch não invocam ações simplesmente por estarem em um estado específico. O estado deve ter sido alterado e mantido por um número específico de períodos.

# Configurar o registro em log do CloudWatch para APIs REST no API Gateway

Para ajudar a depurar problemas relacionados à execução de solicitação ou ao acesso do cliente à sua API, é possível permitir que o Amazon CloudWatch Logs registre chamadas de API em log. Para obter mais informações sobre o CloudWatch, consulte [the section called “Métricas do CloudWatch”](#).

## Formatos de log do CloudWatch para o API Gateway

Há dois tipos de registro de API em logs no CloudWatch: registro de execução e de acesso. No registro de execução, o API Gateway gerencia o CloudWatch Logs. O processo inclui a criação de grupos de log e fluxos de log, além de relatórios aos fluxos de log sobre solicitações e respostas de qualquer autor da chamada.

Os dados registrados em log incluem erros ou rastreamentos de execução (como cargas úteis ou valores de parâmetro de solicitação ou de resposta), dados usados por autorizadores do Lambda (anteriormente conhecidos como autorizadores personalizados), independentemente de as chaves de API serem necessárias ou de os planos de uso estarem ativos e outras informações. O API Gateway edita cabeçalhos de autorização, valores de chave de API e parâmetros de solicitação confidenciais semelhantes dos dados registrados em log.

Para melhorar seu procedimento de segurança, recomendamos que você use o registro em log de execução no nível ERROR ou INFO. Você pode precisar fazer isso para cumprir vários requisitos de conformidade. Para ter mais informações, consulte [Amazon API Gateway controls](#) no Guia do usuário do AWS Security Hub.

Quando você implanta uma API, o API Gateway cria um grupo de logs e registra os streams no grupo de logs. O grupo de logs é chamado seguindo o formato API-Gateway-Execution-Logs\_{rest-api-id}/{stage\_name}. Dentro de cada grupo de logs, os logs são subdivididos em fluxos de log, os quais são ordenados por Last Event Time conforme os dados registrados em log são reportados.

No registro de acessos, você, assim como um desenvolvedor de API, registra quem acessou sua API e como o autor da chamada acessou a API. Você pode criar seu próprio grupo de logs ou escolher um existente que possa ser gerenciado pelo API Gateway. Para especificar os detalhes de acesso, selecione variáveis [\\$context](#), um formato de log e um destino do grupo de logs.



O formato do log de acesso deve incluir ao menos `$context.requestId` ou `$context.extendedRequestId`. Como prática recomendada, inclua `$context.requestId` e `$context.extendedRequestId` em seu formato de log.

### **`$context.requestId`**

Isso registra em log o valor no cabeçalho `x-amzn-RequestId`. Os clientes podem substituir o valor no cabeçalho `x-amzn-RequestId` por um valor no formato de um identificador universal exclusivo (UUID). O API Gateway retorna esse ID de solicitação no cabeçalho de resposta `x-amzn-RequestId`. O API Gateway substitui os IDs de solicitação substituídos que não estão no formato de um UUID com `UUID_REPLACED_INVALID_REQUEST_ID` nos logs de acesso.

### **`$context.extendedRequestId`**

`extendedRequestId` é um ID exclusivo gerado pelo API Gateway. O API Gateway retorna esse ID de solicitação no cabeçalho de resposta `x-amz-apigw-id`. Um autor da chamada de API não pode fornecer ou substituir esse ID de solicitação. Pode ser necessário fornecer esse valor ao suporte da AWS para ajudar a solucionar problemas de sua API. Para obter mais informações, consulte [the section called “Variáveis para registro em log de acesso no API Gateway”](#).

Escolha um formato de log que também seja adotado pelo seu backend de análise, como [Common Log Format](#) (CLF), JSON, XML ou CSV. Em seguida, você pode enviar os logs de acesso a ele diretamente para que suas métricas sejam calculadas e produzidas. Para definir o formato de log, defina o ARN do grupo de logs na propriedade [accessLogSettings/destinationArn](#) no [estágio](#). Você pode obter o ARN de um grupo de logs no console do CloudWatch. Para definir o formato de log de acesso, defina um formato escolhido na propriedade [accessLogSetting/format](#) no [estágio](#).

Exemplos de alguns formatos de log de acesso comumente usados são mostrados no console do API Gateway e estão listados a seguir.

- CLF ([Formato de log comum](#)):

```
$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 vírgula):

```
$context.identity.sourceIp,$context.identity.caller,$context.identity.user,
$context.requestTime,$context.httpMethod,$context.resourcePath,$context.protocol,
$context.status,$context.responseLength,$context.requestId,$context.extendedRequestId
```

## Permissões para registro em log do CloudWatch

Para habilitar o CloudWatch Logs, é necessário conceder permissão ao API Gateway para ler e gravar logs no CloudWatch para sua conta. O [AmazonAPIGatewayPushToCloudWatchLogs](#) tem todas as permissões necessárias.

### Note

O API Gateway chama o AWS Security Token Service para assumir a função do IAM. Portanto, certifique-se de que o AWS STS esteja habilitado para a região. Para obter mais informações, consulte [Gerenciar o AWS STS em uma região da AWS](#).

Para conceder essas permissões à sua conta, crie uma função do IAM com `apigateway.amazonaws.com` como entidade confiável, anexe a política anterior à função do IAM e defina o ARN da função do IAM na propriedade [cloudWatchRoleArn](#) em sua [conta](#). Você deve

definir a propriedade [CloudWatchroLearn](#) separadamente para cada região da AWS na qual deseja habilitar o CloudWatch Logs.

Se você receber um erro ao definir o ARN da função do IAM, verifique as configurações da sua conta da AWS Security Token Service para garantir que o AWS STS esteja habilitado na região que você está usando. Para obter mais informações sobre como ativar o AWS STS, consulte [Gerenciar o AWS STS em uma região da AWS](#) no Guia do usuário do IAM.

## Configurar o registro em log da API do CloudWatch usando o console do API Gateway

Para configurar o registro em log da API do CloudWatch, é necessário ter implantado a API em um estágio. Você também deve ter configurado um [ARN de função do CloudWatch Logs adequado](#) para a sua conta.

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, selecione Configurações e, em Registro em log, selecione Editar.
3. Em ARN de função do log do CloudWatch, insira o ARN de um perfil do IAM com as permissões apropriadas. É necessário fazer isso uma vez para cada Conta da AWS que cria APIs usando o API Gateway.
4. No painel de navegação principal, selecione APIs e siga um destes procedimentos:
  - a. Selecione uma API e escolha um estágio.
  - b. Crie uma API e implante-a em um estágio.
5. No painel de navegação principal, selecione Estágios.
6. Na seção Logs e rastreamento, selecione Editar.
7. Para habilitar o registro de execução em logs:
  - a. Selecione um nível de registro em log no menu suspenso CloudWatch Logs. Os níveis de registro em log são os seguintes:
    - Desativado: o registro em log não está ativado neste estágio.
    - Somente erros: o registro em log está habilitado somente para erros.
    - Logs de erros e informações: o registro em log está habilitado para todos os eventos.
  - b. (Opcional) Selecione Rastreamento de dados para ativar o registro em log de rastreamento de dados para seu estágio. Isso pode ser útil para solucionar problemas de APIs, mas pode resultar no registro de dados confidenciais.

 Note

Recomendamos não usar Rastreamento de dados para APIs de produção.

- c. (Opcional) Selecione Métricas detalhadas para ativar as métricas detalhadas do CloudWatch.

Para obter mais informações sobre métricas do CloudWatch, consulte [the section called “Métricas do CloudWatch”](#).

- 8. Para habilitar o registro de acesso em logs:
  - a. Ative o Registro em log de acesso personalizado.
  - b. Em Acessar ARN de destino do log, insira o ARN de um grupo de logs. O formato do ARN é `arn:aws:logs:{region}:{account-id}:log-group:log-group-name`.
  - c. Em Formato do log, insira um formato de log. É possível escolher CLF, JSON, XML ou CSV. Para saber mais sobre exemplos de formatos de log, consulte [the section called “Formatos de log do CloudWatch para o API Gateway”](#).
- 9. Escolha Salvar alterações.

 Note

Você pode permitir o registro em log de execução e o de acesso independentemente um do outro.

O API Gateway já está pronto para registrar solicitações à sua API em log. Não é necessário reimplantar a API ao atualizar as configurações do estágio, os logs ou as variáveis do estágio.

## Configurar o registro da API do CloudWatch usando o CloudFormation

Use o modelo do CloudFormation de exemplo a seguir para criar um grupo de logs do Amazon CloudWatch Logs e configurar a execução e o registro de acesso para um estágio. Para habilitar o CloudWatch Logs, é necessário conceder permissão ao API Gateway para ler e gravar logs no CloudWatch para sua conta. Para saber mais, consulte [Associar a conta ao perfil do IAM](#) no Guia do usuário do AWS CloudFormation.

```
TestStage:
 Type: AWS::ApiGateway::Stage
 Properties:
 StageName: test
 RestApiId: !Ref MyAPI
 DeploymentId: !Ref Deployment
 Description: "test stage description"
 MethodSettings:
 - ResourcePath: "/*"
 HttpMethod: "*"
 LoggingLevel: INFO
 AccessLogSetting:
 DestinationArn: !GetAtt MyLogGroup.Arn
 Format: $context.extendedRequestId $context.identity.sourceIp
 $context.identity.caller $context.identity.user [$context.requestTime]
 "$context.httpMethod $context.resourcePath $context.protocol" $context.status
 $context.responseLength $context.requestId
 MyLogGroup:
 Type: AWS::Logs::LogGroup
 Properties:
 LogGroupName: !Join
 - '-'
 - - !Ref MyAPI
 - access-logs
```

## Registrar em log chamadas de API REST para o Amazon Data Firehose no API Gateway

Para ajudar a depurar problemas relacionados ao acesso do cliente à sua API, é possível registrar em log chamadas de API no Amazon Data Firehose. Para saber mais sobre o Amazon Firehose, consulte [What Is Amazon Data Firehose?](#).

Para o registro em log de acesso, você só pode habilitar o CloudWatch ou o Firehose, não os dois. No entanto, você pode habilitar o CloudWatch para registro em log de execução e o Firehose para registro em log de acesso.

### Tópicos

- [Formatos de log do Firehose para o API Gateway](#)
- [Permissões para registro em log do Firehose](#)

- [Configurar o registro em log de acesso do Firehose usando o console do API Gateway](#)

## Formatos de log do Firehose para o API Gateway

O registro em log do Firehose usa o mesmo formato que o [registro em log do CloudWatch](#).

## Permissões para registro em log do Firehose

Quando o registro em log de acesso do Firehose estiver habilitado em um estágio, o API Gateway criará um perfil vinculado ao serviço na sua conta, caso o perfil ainda não exista. A função será chamada de `AWSServiceRoleForAPIGateway` e terá a política gerenciada `APIGatewayServiceRolePolicy` anexada à ela. Para obter mais informações sobre funções vinculadas a um serviço, consulte [Como usar funções vinculadas a serviços](#).

### Note

O nome do fluxo do Firehose deve ser `amazon-apigateway-{your-stream-name}`.

## Configurar o registro em log de acesso do Firehose usando o console do API Gateway

Para configurar o registro de API em logs, você deve ter implantado a API em um estágio. Também é necessário ter criado um fluxo do Firehose.

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Execute um destes procedimentos:
  - a. Selecione uma API e escolha um estágio.
  - b. Crie uma API e implante-a em um estágio.
3. No painel de navegação principal, selecione Estágios.
4. Na seção Logs e rastreamento, selecione Editar.
5. Para habilitar o registro em log de acesso a um fluxo do Firehose:
  - a. Ative o Registro em log de acesso personalizado.
  - b. Em ARN de destino de logs de acesso, insira o ARN de um fluxo do Firehose. O formato do ARN é `arn:aws:firehose:{region}:{account-id}:deliverystream/amazon-apigateway-{your-stream-name}`.

**Note**

O nome do fluxo do Firehose deve ser `amazon-apigateway-{your-stream-name}`.

- c. Em Formato do log, insira um formato de log. É possível escolher CLF, JSON, XML ou CSV. Para saber mais sobre exemplos de formatos de log, consulte [the section called “Formatos de log do CloudWatch para o API Gateway”](#).

6. Escolha Salvar alterações.

Agora o API Gateway está pronto para registrar em log no Firehose solicitações feitas à API. Não é necessário reimplantar a API ao atualizar as configurações do estágio, os logs ou as variáveis do estágio.

## Variáveis para registro em log de acesso no API Gateway

No registro de acessos, você, assim como um desenvolvedor de API, registra quem acessou sua API e como o autor da chamada acessou a API. Você pode criar seu próprio grupo de logs ou escolher um existente que possa ser gerenciado pelo API Gateway. Para especificar os detalhes de acesso, você pode usar as variáveis `$context` com distinção de maiúsculas e minúsculas a seguir.

Consulte uma lista de variáveis de referência para transformadores de dados em [the section called “Variáveis para transformações de dados”](#).

| Parameter                                | Descrição                                                     |
|------------------------------------------|---------------------------------------------------------------|
| <code>\$context.accountId</code>         | O ID da conta da AWS do proprietário da API                   |
| <code>\$context.apiId</code>             | O identificador que o API Gateway atribui à sua API.          |
| <code>\$context.authorize.error</code>   | A mensagem de erro de autorização.                            |
| <code>\$context.authorize.latency</code> | A latência de autorização em ms.                              |
| <code>\$context.authorize.status</code>  | O código de status retornado de uma tentativa de autorização. |

| Parameter                                                     | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|---------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$context.authorizer.claims.<br/><i>property</i></code> | <p>Uma propriedade das declarações retornadas do grupo de usuários do Amazon Cognito depois que o autor da chamada do método é autenticado com êxito. Para obter mais informações, consulte <a href="#">the section called “Use o grupo de usuários do Amazon Cognito como um autorizador para uma API REST”</a>.</p> <div>  <b>Note</b><br/>           Chamar <code>\$context.authorizer.claims</code> retorna um valor nulo.         </div> |
| <code>\$context.authorizer.error</code>                       | A mensagem de erro retornada de um autorizador.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <code>\$context.authorizer.integrationLatency</code>          | A latência de integração do autorizador em ms.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <code>\$context.authorizer.integrationStatus</code>           | O código de status retornado de um autorizador do Lambda.                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <code>\$context.authorizer.latency</code>                     | A latência de autorizador em ms.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <code>\$context.authorizer.principalId</code>                 | A identificação do usuário principal associada ao token enviado pelo cliente e retornada a partir de um autorizador do Lambda do API Gateway (anteriormente conhecido como autorizador personalizado). Para obter mais informações, consulte <a href="#">the section called “Usar os autorizadores do Lambda”</a> .                                                                                                                                                                                                            |



| Parameter                                          | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$context.authorizer.</code> <i>property</i> | <p>O valor transformado em string do par de chave/valor especificado do mapa context retornado de uma função de autorizador do Lambda do API Gateway. Por exemplo, se o autorizador retornar o seguinte mapa context:</p> <pre>"context" : {<br/>  "key": "value",<br/>  "numKey": 1,<br/>  "boolKey": true<br/>}</pre> <p>Chamar <code>\$context.authorizer.key</code> retornará a string "value", chamar <code>\$context.authorizer.numKey</code> retornará a string "1" e chamar <code>\$context.authorizer.boolKey</code> retornará a string "true".</p> <p>Com relação à <i>propriedade</i> , o único caractere especial aceito é o sublinhado (<code>_</code>).</p> <p>Para obter mais informações, consulte <a href="#">the section called “Usar os autorizadores do Lambda”</a>.</p> |
| <code>\$context.authorizer.requestId</code>        | O ID da solicitação do endpoint da AWS.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <code>\$context.authorizer.status</code>           | O código de status retornado de um autorizador.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <code>\$context.authenticate.error</code>          | A mensagem de erro retornada de uma tentativa de autenticação.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <code>\$context.authenticate.latency</code>        | A latência de autenticação em ms.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

| Parameter                                                | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$context.authenticate.status</code>               | O código de status retornado de uma tentativa de autenticação.                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <code>\$context.awsEndpointRequestId</code>              | O ID da solicitação do endpoint da AWS.                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <code>\$context.cipherSuite</code>                       | A criptografia, no formato IANA, que é negociada durante o handshake TLS entre o cliente e o API Gateway.                                                                                                                                                                                                                                                                                                                                                                                             |
| <code>\$context.customDomain.basePathMatched</code>      | O caminho para um mapeamento da API correspondente a uma solicitação de entrada. Aplicável quando um cliente usa um nome de domínio personalizado para acessar uma API. Por exemplo, se um cliente enviar uma solicitação para <code>https://api.example.com/v1/orders/1234</code> e a solicitação corresponder ao mapeamento da API com o caminho <code>v1/orders</code> , o valor será <code>v1/orders</code> . Para saber mais, consulte <a href="#">the section called “Mapeamentos de API”</a> . |
| <code>\$context.customDomain.routingRuleIdMatched</code> | A regra de roteamento atendida por uma solicitação de entrada. Aplicável quando um cliente usa um nome de domínio personalizado para acessar uma API. Para saber mais, consulte <a href="#">the section called “Regras de roteamento”</a> .                                                                                                                                                                                                                                                           |
| <code>\$context.deploymentId</code>                      | O ID da implantação de API.                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <code>\$context.domainName</code>                        | O nome de domínio completo usado para invocar a API. Ele deve ser o mesmo que o cabeçalho Host de entrada.                                                                                                                                                                                                                                                                                                                                                                                            |
| <code>\$context.domainPrefix</code>                      | O primeiro rótulo do <code>\$context.domainName</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                              |

| Parameter                                          | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$context.endpointType</code>                | O tipo do endpoint da API.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <code>\$context.error.message</code>               | Uma string que contém uma mensagem de erro do API Gateway. Essa variável pode ser usada apenas para substituição de variável simples em um modelo de mapeamento de corpo <a href="#">GatewayResponse</a> , que não é processado pelo mecanismo Velocity Template Language, bem como no registro de acesso. Para obter mais informações, consulte <a href="#">the section called “Métricas”</a> e <a href="#">the section called “Configurar respostas do gateway para personalizar respostas de erro”</a> .   |
| <code>\$context.error.messageString</code>         | O valor citado de <code>\$context.error.message</code> , ou seja, " <code>\$context.error.message</code> ".                                                                                                                                                                                                                                                                                                                                                                                                   |
| <code>\$context.error.responseType</code>          | Um <a href="#">tipo</a> de <a href="#">GatewayResponse</a> . Essa variável pode ser usada apenas para substituição de variável simples em um modelo de mapeamento de corpo <a href="#">GatewayResponse</a> , que não é processado pelo mecanismo Velocity Template Language, bem como no registro de acesso. Para obter mais informações, consulte <a href="#">the section called “Métricas”</a> e <a href="#">the section called “Configurar respostas do gateway para personalizar respostas de erro”</a> . |
| <code>\$context.error.validationErrorString</code> | Uma string que contém uma mensagem de erro de validação detalhada.                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <code>\$context.extendedRequestId</code>           | O ID estendido que o API Gateway gera e atribui à solicitação de API. O ID de solicitação estendido contém informações úteis para a depuração e solução de problemas.                                                                                                                                                                                                                                                                                                                                         |

| Parameter                                 | Descrição                                                                                                                                                                                                                                                                                     |
|-------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$context.httpMethod</code>         | O método HTTP utilizado. Os valores válidos incluem: DELETE, GET, HEAD, OPTIONS, PATCH, POST e PUT.                                                                                                                                                                                           |
| <code>\$context.identity.accountId</code> | O ID da conta da AWS associado à solicitação.                                                                                                                                                                                                                                                 |
| <code>\$context.identity.apiKey</code>    | Para os métodos de API que exigem uma chave de API, essa variável é a chave de API associada à solicitação do método. Para métodos que não exigem uma chave de API, essa variável é um valor nulo. Para obter mais informações, consulte <a href="#">the section called “Planos de uso”</a> . |
| <code>\$context.identity.apiKeyId</code>  | O ID da chave de API associada a uma solicitação de API que exige uma chave de API.                                                                                                                                                                                                           |
| <code>\$context.identity.caller</code>    | O identificador principal da entidade do autor da chamada que assinou a solicitação. Compatível com recursos que usam a autorização do IAM.                                                                                                                                                   |

| Parameter                                                     | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|---------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$context.identity.cognitoAuthenticationProvider</code> | <p>Uma lista separada por vírgulas de todos os provedores de autenticação do Amazon Cognito usados pelo autor da chamada que faz a solicitação. Disponível somente se a solicitação foi assinada com credenciais do Amazon Cognito.</p> <p>Por exemplo, para uma identidade e de um grupo de usuários do Amazon Cognito, <code>cognito-idp.<i>region</i>.amazonaws.com/<i>user_pool_id</i></code>, <code>cognito-idp.<i>region</i>.amazonaws.com/<i>user_pool_id</i>:CognitoSignIn:<i>token subject claim</i></code></p> <p>Consulte informações sobre os provedores de autenticação do Amazon Cognito disponíveis em <a href="#">Using Federated Identities</a> no Guia do desenvolvedor do Amazon Cognito.</p> |
| <code>\$context.identity.cognitoAuthenticationType</code>     | <p>O tipo de autenticação do Amazon Cognito do autor da chamada que faz a solicitação. Disponível somente se a solicitação foi assinada com credenciais do Amazon Cognito. Os valores possíveis incluem <code>authenticated</code> para identidades autenticadas e <code>unauthenticated</code> para identidades não autenticadas.</p>                                                                                                                                                                                                                                                                                                                                                                           |
| <code>\$context.identity.cognitoIdentityId</code>             | <p>O ID de identidade do Amazon Cognito do autor da chamada que faz a solicitação. Disponível somente se a solicitação foi assinada com credenciais do Amazon Cognito.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

| Parameter                                                                  | Descrição                                                                                                                                                                                                                                                                                                 |
|----------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$context.identity.cognitoId</code><br><code>entityPoolId</code>     | O ID do grupo de identidades do Amazon Cognito do autor da chamada que faz a solicitação. Disponível somente se a solicitação foi assinada com credenciais do Amazon Cognito.                                                                                                                             |
| <code>\$context.identity.principalOrgId</code>                             | O <a href="#">ID da organização da AWS</a> .                                                                                                                                                                                                                                                              |
| <code>\$context.identity.sourceIp</code>                                   | O endereço IP de origem da conexão TCP mais próxima que está fazendo a solicitação ao endpoint do API Gateway.                                                                                                                                                                                            |
| <code>\$context.identity.clientCert</code><br><code>t.clientCertPem</code> | O certificado de cliente codificado por PEM que o cliente apresentou durante a autenticação TLS mútua. Presente quando um cliente acessa uma API usando um nome de domínio personalizado que tenha TLS mútuo habilitado. Presente somente nos logs de acesso, se ocorrer falha na autenticação TLS mútua. |
| <code>\$context.identity.clientCert</code><br><code>t.subjectDN</code>     | O nome distinto do assunto do certificado apresentado por um cliente. Presente quando um cliente acessa uma API usando um nome de domínio personalizado que tenha TLS mútuo habilitado. Presente somente nos logs de acesso, se ocorrer falha na autenticação TLS mútua.                                  |
| <code>\$context.identity.clientCert</code><br><code>t.issuerDN</code>      | O nome distinto do emissor do certificado apresentado por um cliente. Presente quando um cliente acessa uma API usando um nome de domínio personalizado que tenha TLS mútuo habilitado. Presente somente nos logs de acesso, se ocorrer falha na autenticação TLS mútua.                                  |

| Parameter                                                            | Descrição                                                                                                                                                                                                                                         |
|----------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$context.identity.clientCertificate.serialNumber</code>       | O número de série do certificado. Presente quando um cliente acessa uma API usando um nome de domínio personalizado que tenha TLS mútuo habilitado. Presente somente nos logs de acesso, se ocorrer falha na autenticação TLS mútua.              |
| <code>\$context.identity.clientCertificate.validity.notBefore</code> | A data antes da qual o certificado é inválido. Presente quando um cliente acessa uma API usando um nome de domínio personalizado que tenha TLS mútuo habilitado. Presente somente nos logs de acesso, se ocorrer falha na autenticação TLS mútua. |
| <code>\$context.identity.clientCertificate.validity.notAfter</code>  | A data após a qual o certificado é inválido. Presente quando um cliente acessa uma API usando um nome de domínio personalizado que tenha TLS mútuo habilitado. Presente somente nos logs de acesso, se ocorrer falha na autenticação TLS mútua.   |
| <code>\$context.identity.vpcId</code>                                | O ID da VPC que está fazendo a solicitação ao endpoint do API Gateway.                                                                                                                                                                            |
| <code>\$context.identity.vpceId</code>                               | O ID do endpoint da VPC que está fazendo a solicitação ao endpoint do API Gateway. Presente somente quando você tem uma API privada.                                                                                                              |
| <code>\$context.identity.user</code>                                 | O identificador principal da entidade do usuário que será autorizado contra o acesso a recursos. Compatível com recursos que usam a autorização do IAM.                                                                                           |
| <code>\$context.identity.userAgent</code>                            | O cabeçalho <a href="#">User-Agent</a> do autor da chamada da API.                                                                                                                                                                                |

| Parameter                                               | Descrição                                                                                                                                                                                                                                                               |
|---------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$context.identity.userArn</code>                 | O Nome do Recurso Amazon (ARN) do usuário efetivo identificado após a autenticação. Para obter mais informações, consulte <a href="https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html">https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html</a> . |
| <code>\$context.integration.error</code>                | A mensagem de erro retornada de uma integração.                                                                                                                                                                                                                         |
| <code>\$context.integration.integrationStatus</code>    | Para a integração de proxy do Lambda, o código de status retornado pelo AWS Lambda, e não pelo código de função do Lambda de backend.                                                                                                                                   |
| <code>\$context.integration.latency</code>              | A latência de integração em ms. Equivale a <code>\$context.integrationLatency</code> .                                                                                                                                                                                  |
| <code>\$context.integration.requestId</code>            | O ID da solicitação do endpoint da AWS. Equivale a <code>\$context.awsEndpointRequestId</code> .                                                                                                                                                                        |
| <code>\$context.integration.responseTransferMode</code> | O modo de transferência de resposta da integração. Ele pode ser BUFFERED ou STREAMED.                                                                                                                                                                                   |
| <code>\$context.integration.status</code>               | O código de status retornado de uma integração. Para integrações de proxy do Lambda, esse é o código de status que seu código de função do Lambda retorna.                                                                                                              |
| <code>\$context.integration.timeToAllHeaders</code>     | O tempo entre o momento em que o API Gateway estabelece a conexão de integração e o momento em que recebe todos os cabeçalhos de resposta de integração do cliente.                                                                                                     |



| Parameter                                             | Descrição                                                                                                                                                                                                                                        |
|-------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$context.integration.timeToFirstContent</code> | O tempo entre o momento em que o API Gateway estabelece a conexão de integração e o momento em que recebe os primeiros bytes de conteúdo.                                                                                                        |
| <code>\$context.integrationLatency</code>             | A latência de integração em ms.                                                                                                                                                                                                                  |
| <code>\$context.integrationStatus</code>              | Para a integração de proxy do Lambda, esse parâmetro representa o código de status retornado do AWS Lambda, não do código da função do Lambda de backend.                                                                                        |
| <code>\$context.isCanaryRequest</code>                | Retorna <code>true</code> se a solicitação foi direcionada ao canário e <code>false</code> se a solicitação não foi direcionada ao canário. Presente somente quando você tem um canário habilitado.                                              |
| <code>\$context.path</code>                           | O caminho da solicitação. Por exemplo, para um URL de solicitação sem proxy de <code>https://{rest-api-id}.execute-api.{region}.amazonaws.com/{stage}/root/child</code> , o valor <code>\$context.path</code> é <code>/stage/root/child</code> . |

| Parameter                                                         | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$context.protocol</code>                                   | <p>O protocolo de solicitação, por exemplo, , HTTP/1.1.</p> <div> <b>Note</b><br/>As APIs do API Gateway podem aceitar solicitações HTTP/2, mas o API Gateway envia solicitações para integrações de back-end usando HTTP/1.1. Como resultado, o protocolo de solicitação é registrado como HTTP/1.1 mesmo se um cliente enviar uma solicitação que usa HTTP/2.</div> |
| <code>\$context.requestId</code>                                  | <p>Um ID para a solicitação. Os clientes podem substituir esse ID de solicitação. Usar <code>\$context.extendedRequestId</code> para um ID de solicitação exclusivo gerado pelo API Gateway.</p>                                                                                                                                                                                                                                                       |
| <code>\$context.requestOverride.header. <i>header_name</i></code> | <p>Substituição do cabeçalho da solicitação. Esse parâmetro, quando definido, contém os cabeçalhos a serem usados em lugar dos HTTP Headers (Cabeçalhos HTTP) que são definidos no painel Integration Request (Solicitação de integração). Para obter mais informações, consulte <a href="#">Substituir parâmetros de solicitação e resposta da API e códigos de status por APIs REST no API Gateway</a>.</p>                                          |

| Parameter                                                                   | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-----------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$context.requestOverride.path.</code> <i>path_name</i>               | Substituição do caminho da solicitação. Esse parâmetro, quando definido, contém o caminho da solicitação a ser usado em lugar dos URL Path Parameters (Parâmetros de caminho de URL) que são definidos no painel Integration Request (Solicitação de integração). Para obter mais informações, consulte <a href="#">Substituir parâmetros de solicitação e resposta da API e códigos de status por APIs REST no API Gateway</a> .                                               |
| <code>\$context.requestOverride.querystring.</code> <i>querystring_name</i> | Substituição da string de consulta da solicitação. Esse parâmetro, quando definido, contém as strings de consulta da solicitação a serem usadas em lugar dos URL Query String Parameters (Parâmetros de string de consulta de URL) que são definidos no painel Integration Request (Solicitação de integração). Para obter mais informações, consulte <a href="#">Substituir parâmetros de solicitação e resposta da API e códigos de status por APIs REST no API Gateway</a> . |
| <code>\$context.responseLatency</code>                                      | A latência da resposta em ms.                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <code>\$context.responseLength</code>                                       | O tamanho da carga de resposta em bytes.                                                                                                                                                                                                                                                                                                                                                                                                                                        |

| Parameter                                                          | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|--------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$context.responseOverride.header. <i>header_name</i></code> | Substituição do cabeçalho da resposta. Esse parâmetro, quando definido, contém o cabeçalho a ser retornado em lugar do Response header (Cabeçalho de resposta) que é definido como o Default mapping (Mapeamento padrão) no painel Integration Response (Resposta de integração). Para obter mais informações, consulte <a href="#">Substituir parâmetros de solicitação e resposta da API e códigos de status por APIs REST no API Gateway</a> .                             |
| <code>\$context.responseOverride.status</code>                     | Substituição do código de status da resposta. Esse parâmetro, quando definido, contém o código de status a ser retornado em lugar do Method response status (Status de resposta de método) que é definido como o Default mapping (Mapeamento padrão) no painel Integration Response (Resposta de integração). Para obter mais informações, consulte <a href="#">Substituir parâmetros de solicitação e resposta da API e códigos de status por APIs REST no API Gateway</a> . |
| <code>\$context.requestTime</code>                                 | O horário da solicitação <a href="#">CLF</a> formatado (dd/MMM/yyyy:HH:mm:ss +-hhmm ).                                                                                                                                                                                                                                                                                                                                                                                        |
| <code>\$context.requestTimeEpoch</code>                            | O tempo de solicitação formatado em <a href="#">Epoch</a> , em milissegundos.                                                                                                                                                                                                                                                                                                                                                                                                 |
| <code>\$context.resourceId</code>                                  | O identificador que o API Gateway atribui ao seu recurso.                                                                                                                                                                                                                                                                                                                                                                                                                     |

| Parameter                              | Descrição                                                                                                                                                                                                                                                                                                                                                                         |
|----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$context.resourcePath</code>    | O caminho para o seu recurso. Por exemplo, para a URI de solicitação sem proxy de <code>https://{rest-api-id}.execute-api.{region}.amazonaws.com/{stage}/root/child</code> , o valor <code>\$context.resourcePath</code> é <code>/root/child</code> . Para obter mais informações, consulte <a href="#">Tutorial: Crie uma API REST com uma integração de não proxy de HTTP</a> . |
| <code>\$context.stage</code>           | O estágio de implantação da solicitação de API (por exemplo, Beta ou Prod).                                                                                                                                                                                                                                                                                                       |
| <code>\$context.status</code>          | O status de resposta do método.                                                                                                                                                                                                                                                                                                                                                   |
| <code>\$context.tlsVersion</code>      | A versão de TLS que é negociada durante o handshake TLS entre o cliente e o API Gateway.                                                                                                                                                                                                                                                                                          |
| <code>\$context.waf.error</code>       | A mensagem de erro retornada pelo AWS WAF.                                                                                                                                                                                                                                                                                                                                        |
| <code>\$context.waf.latency</code>     | A latência do AWS WAF em ms.                                                                                                                                                                                                                                                                                                                                                      |
| <code>\$context.waf.status</code>      | O código de status retornado pelo AWS WAF.                                                                                                                                                                                                                                                                                                                                        |
| <code>\$context.xrayTraceId</code>     | O ID de rastreamento do rastreamento do X-Ray. Para obter mais informações, consulte <a href="#">the section called “Configurar o AWS X-Ray”</a> .                                                                                                                                                                                                                                |
| <code>\$context.wafResponseCode</code> | A resposta recebida do <a href="#">AWS WAF</a> : <code>WAF_ALLOW</code> ou <code>WAF_BLOCK</code> . Não será definido se o estágio não estiver associado a uma ACL da web. Para obter mais informações, consulte <a href="#">the section called “AWS WAF”</a> .                                                                                                                   |

| Parameter                        | Descrição                                                                                                                                                                                                                                                           |
|----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$context.webaclArn</code> | O ARN completo da ACL da web que é utilizada para decidir se deseja permitir ou bloquear a solicitação. Não será definido se o estágio não estiver associado a uma ACL da web. Para obter mais informações, consulte <a href="#">the section called “AWS WAF”</a> . |

## Rastrear solicitações de usuário para APIs REST usando o X-Ray no API Gateway

Você pode usar o [AWS X-Ray](#) para rastrear e analisar solicitações de usuário à medida que passam pelas APIs REST do Amazon API Gateway a caminho dos serviços subjacentes. O API Gateway oferece suporte ao rastreamento de do X-Ray para todos os tipos de endpoint de API REST do API Gateway: regional, otimizado para bordas e privado. Você pode usar o X-Ray com o Amazon API Gateway em todas as regiões da AWS em que o X-Ray está disponível.

Como o X-Ray fornece uma visão completa de uma solicitação inteira, você pode analisar latências em suas APIs e serviços de backend. Use um mapa de serviço do X-Ray para visualizar a latência de uma solicitação inteira e dos serviços downstream integrados ao X-Ray. Também é possível configurar regras de amostragem para informar ao X-Ray quais solicitações registrar e com quais taxas de amostragem, de acordo com os critérios especificados.

Se você chamar uma API do API Gateway de um serviço que já está sendo rastreado, o API Gateway propagará o rastreamento, mesmo que o rastreamento do X-Ray não esteja habilitado na API.

Você pode habilitar o X-Ray para um estágio da API usando o console do API Gateway ou usando a CLI ou a API do API Gateway.

### Tópicos

- [Configurar o AWS X-Ray com APIs REST do API Gateway](#)
- [Usar mapas de serviço e visualizações de rastreamento do AWS X-Ray com o API Gateway](#)
- [Configurar regras de amostragem do AWS X-Ray para APIs do API Gateway](#)
- [Rastreamentos do AWS X-Ray para APIs do Amazon API Gateway](#)

## Configurar o AWS X-Ray com APIs REST do API Gateway

Nesta seção, você encontrará informações detalhadas sobre como configurar o [AWS X-Ray](#) com APIs REST do API Gateway.

### Tópicos

- [Modos de rastreamento do X-Ray para o API Gateway](#)
- [Permissões para rastreamento do X-Ray](#)
- [Habilitar o rastreamento do X-Ray no console do API Gateway](#)
- [Habilitar o rastreamento do AWS X-Ray usando a CLI do API Gateway](#)

### Modos de rastreamento do X-Ray para o API Gateway

O caminho de uma solicitação pelo seu aplicativo é controlado com um ID de rastreamento. Um rastreamento coleta todos os segmentos gerados por uma única solicitação, geralmente uma solicitação HTTP GET ou POST.

Existem dois modos de rastreamento para uma API do API Gateway:

- Passivo: essa será a configuração padrão se o rastreamento do X-Ray não estiver habilitado em um estágio da API. Essa abordagem significa que a API do API Gateway só é rastreada se o X-Ray estiver habilitado em um serviço upstream.
- Ativo: quando um estágio da API do API Gateway tem esta configuração, o API Gateway faz a amostragem automaticamente de solicitações de invocação da API, com base no algoritmo de amostragem especificado pelo X-Ray.

Quando o rastreamento ativo estiver habilitado em um estágio, o API Gateway cria uma função vinculada ao serviço na sua conta, caso a função ainda não exista. A função é chamada de `AWSServiceRoleForAPIGateway` e terá a política gerenciada `APIGatewayServiceRolePolicy` anexada à ela. Para obter mais informações sobre funções vinculadas a um serviço, consulte [Como usar funções vinculadas a serviços](#).

#### Note

O X-Ray aplica um algoritmo de amostragem para garantir que o rastreamento seja eficiente, enquanto ainda fornece uma amostra representativa das solicitações recebidas

pela API. O algoritmo de amostragem padrão é uma solicitação por segundo, com 5% de solicitações de amostra fora do limite.

Você pode alterar o modo de rastreamento da sua API usando o console de gerenciamento do API Gateway, a CLI do API Gateway ou um SDK da AWS.

### Permissões para rastreamento do X-Ray

Quando você habilita o rastreamento do X-Ray em um estágio, o API Gateway cria uma função vinculada ao serviço na sua conta, caso a função ainda não exista. A função é chamada de `AWSServiceRoleForAPIGateway` e terá a política gerenciada `APIGatewayServiceRolePolicy` anexada à ela. Para obter mais informações sobre funções vinculadas a um serviço, consulte [Como usar funções vinculadas a serviços](#).

### Habilitar o rastreamento do X-Ray no console do API Gateway

Use o console do Amazon API Gateway para habilitar o rastreamento ativo em um estágio da API.

Estas instruções presumem que você já implantou a API em um estágio.

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha a API e, no painel de navegação principal, selecione Estágios.
3. No painel Estágios, selecione um estágio.
4. Na seção Logs e rastreamento, selecione Editar.
5. Para habilitar o rastreamento ativo com X-Ray, selecione Habilitar rastreamento com X-Ray para ativar o rastreamento com X-Ray.
6. Escolha Salvar alterações.

Assim que você tiver habilitado o X-Ray para o estágio da API, use o console de gerenciamento do X-Ray para visualizar os rastreamentos e os mapas de serviço.

### Habilitar o rastreamento do AWS X-Ray usando a CLI do API Gateway

O comando [create-stage](#) indicado a abaixo cria um estágio com o rastreamento do X-Ray ativo:

```
aws apigateway create-stage \
 --rest-api-id rest-api-id \
 --stage-name stage-name \
 --tracing-enabled
```



```
--stage-name stage-name \
--deployment-id deployment-id \
--region region \
--tracing-enabled=true
```

A saída será exibida da seguinte forma:

```
{
 "tracingEnabled": true,
 "stageName": stage-name,
 "cacheClusterEnabled": false,
 "cacheClusterStatus": "NOT_AVAILABLE",
 "deploymentId": deployment-id,
 "lastUpdatedDate": 1533849811,
 "createdDate": 1533849811,
 "methodSettings": {}
}
```

O comando [create-stage](#) indicado abaixo cria um estágio sem o rastreamento do X-Ray ativo:

```
aws apigateway create-stage \
 --rest-api-id rest-api-id \
 --stage-name stage-name \
 --deployment-id deployment-id \
 --region region \
 --tracing-enabled=false
```

A saída será exibida da seguinte forma:

```
{
 "tracingEnabled": false,
 "stageName": stage-name,
 "cacheClusterEnabled": false,
 "cacheClusterStatus": "NOT_AVAILABLE",
 "deploymentId": deployment-id,
 "lastUpdatedDate": 1533849811,
 "createdDate": 1533849811,
 "methodSettings": {}
}
```

O comando [update-stage](#) indicado abaixo ativa o rastreamento do X-Ray para uma API implantada:

```
aws apigateway update-stage \
 --rest-api-id rest-api-id \
 --stage-name stage-name \
 --patch-operations op=replace,path=/tracingEnabled,value=true
```

O comando [update-stage](#) indicado abaixo desativa o rastreamento do X-Ray para uma API implantada:

```
aws apigateway update-stage \
 --rest-api-id rest-api-id \
 --stage-name stage-name \
 --region region \
 --patch-operations op=replace,path=/tracingEnabled,value=false
```

A saída será exibida da seguinte forma:

```
{
 "tracingEnabled": false,
 "stageName": stage-name,
 "cacheClusterEnabled": false,
 "cacheClusterStatus": "NOT_AVAILABLE",
 "deploymentId": deployment-id,
 "lastUpdatedDate": 1533850033,
 "createdDate": 1533849811,
 "methodSettings": {}
}
```

Assim que você tiver habilitado o X-Ray para o estágio da sua API, use a CLI do X-Ray para recuperar as informações de rastreamento. Consulte mais informações em [Using the X-Ray API with the AWS CLI](#).

## Usar mapas de serviço e visualizações de rastreamento do AWS X-Ray com o API Gateway

Nesta seção, encontre informações detalhadas sobre como usar mapas de serviço e exibições de rastreamento do [AWS X-Ray](#) com o API Gateway.

### Tópicos

- [Exemplo de mapa de serviço do X-Ray](#)
- [Exemplo de exibição de rastreamento do X-Ray](#)

## Exemplo de mapa de serviço do X-Ray

AWS X-Ray Os mapas de serviço do mostram informações sobre a API e todos seus serviços downstream. Quando o X-Ray estiver habilitado para um estágio da API no API Gateway, você verá um nó no mapa de serviço contendo informações sobre o tempo total gasto no serviço API Gateway. Obtenha informações detalhadas sobre o status de resposta e um histograma do tempo de resposta da API para o período selecionado. Para APIs que integram-se a serviços da AWS, como o AWS Lambda e o Amazon DynamoDB, você verá mais nós que fornecem métricas de performance relacionadas a esses serviços. Haverá um mapa de serviço para cada estágio da API.

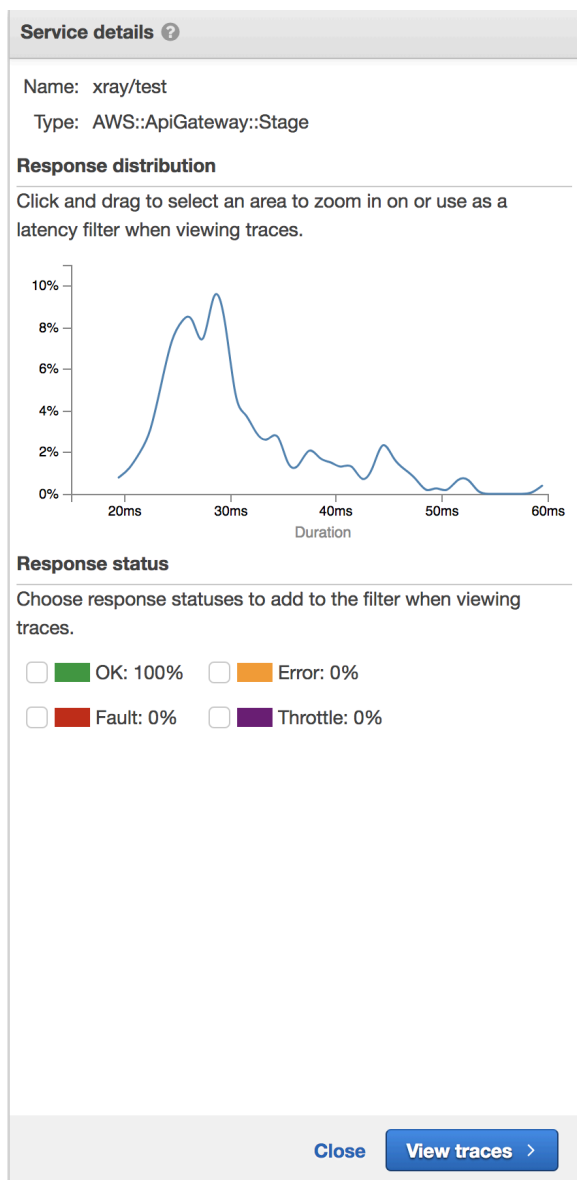
O exemplo a seguir mostra um mapa de serviço para o estágio test de uma API chamada xray. Essa API tem duas integrações do Lambda. Os nós representam o serviço do API Gateway e as duas funções do Lambda.

Consulte uma explicação detalhada sobre a estrutura do mapa de serviço em [Use the X-Ray trace map](#).



No mapa de serviço, você pode ampliar para ver uma exibição de rastreamento do estágio da API. O rastreamento exibirá informações mais detalhadas sobre a API, representadas como segmentos e subsegmentos. Por exemplo, o rastreamento para o mapa de serviço mostrado acima incluiria segmentos para o serviço Lambda e a função do Lambda. Para obter mais informações, consulte [AWS Lambda e AWS X-Ray](#).

Se você escolher um nó ou uma borda em um mapa de serviço do X-Ray, o console do X-Ray mostrará um histograma de distribuição da latência. Você pode usar um histograma de latência para ver o tempo necessário para que um serviço conclua suas solicitações. Veja a seguir um histograma do estágio do API Gateway chamado `xray/test` no mapa de serviço anterior. Consulte uma explicação detalhada sobre os histogramas de distribuição de latência em [Use Latency Histograms](#).

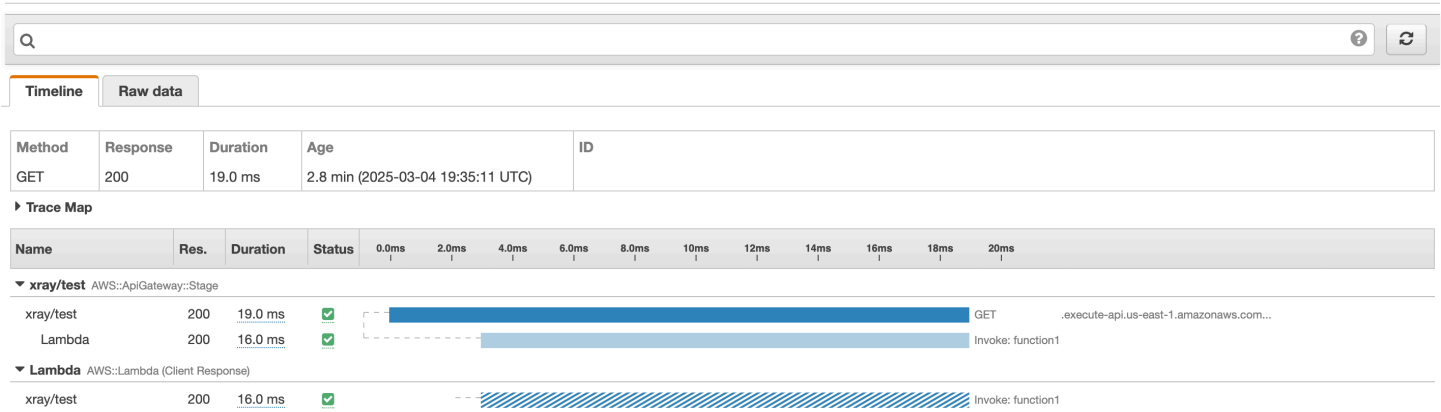


## Exemplo de exibição de rastreamento do X-Ray

O diagrama a seguir mostra uma visualização de rastreamento gerada para a API de exemplo descrita acima, com uma função de backend do Lambda. Uma solicitação de método da API bem-sucedida é mostrada com um código de resposta de 200.

Consulte uma explicação detalhada sobre visualizações de rastreamento em [View traces and trace details](#).

Traces > Details



## Configurar regras de amostragem do AWS X-Ray para APIs do API Gateway

Use o console ou o SDK do AWS X-Ray para configurar regras de amostragem para a API do Amazon API Gateway. Uma regra de amostragem especifica quais solicitações o X-Ray deve registrar para sua API. Ao personalizar regras de amostragem, você pode controlar a quantidade de dados gravados e modificar o comportamento de amostragem instantaneamente, sem modificar ou reimplantar seu código.

Antes de especificar as regras de amostragem do X-Ray, leia os tópicos a seguir no Guia do desenvolvedor do X-Ray:

- [Configure sampling rules](#)
- [Usar regras de amostragem com a API do X-Ray](#)

### Tópicos

- [Valores de opção da regra de amostragem do X-Ray para APIs do API Gateway](#)
- [Exemplos de regras de amostragem do X-Ray](#)

## Valores de opção da regra de amostragem do X-Ray para APIs do API Gateway

As seguintes opções de amostragem do X-Ray são relevantes para o API Gateway. Valores de string podem usar curingas para corresponder a um caractere único (?), ou zero ou mais caracteres (\*). Consulte mais detalhes, incluindo uma explicação detalhada de como as configurações de Reservatório e Taxa são usadas, em [Configure sampling rules](#).

- Nome da regra (string): um nome exclusivo para a regra.
- Prioridade (inteiro entre 1 e 9999): a prioridade da regra de amostragem. Os serviços avaliam as regras em ordem decrescente de prioridade e tomam uma decisão de amostragem com a primeira regra correspondente.
- Reservatório (inteiro não negativo): um número fixo de solicitações correspondentes para instrumentar por segundo, antes de aplicar a taxa fixa. O reservatório não é usado diretamente pelos serviços, mas se aplica a todos os serviços usando a regra coletivamente.
- Taxa (número entre 0 e 100): a porcentagem de solicitações correspondentes para instrumentar, depois que o reservatório é esgotado.
- Nome do serviço (string): nome de estágio da API, no formato **{api-name}/{stage-name}**. Por exemplo, se você implantaria a amostra de API [PetStore](#) em um estágio chamado test, o valor Service name (Nome de serviço) a ser especificado na regra de amostragem seria **pets/test**.
- Tipo de serviço (string): para uma API do API Gateway, pode-se especificar **AWS::ApiGateway::Stage** ou **AWS::ApiGateway::\***.
- Host (string): o nome de host do cabeçalho de host HTTP. Defina isso como \* para corresponder contra todos os nomes de host. Ou especifique um nome de host completo ou parcial para correspondência, por exemplo, **api.example.com** ou **\*.example.com**.
- ARN do recurso (string): o ARN do estágio da API, por exemplo, **arn:aws:apigateway:region::/restapis/api-id/stages/stage-name**.

O nome de estágio pode ser obtido do console, da CLI ou da API do API Gateway. Para obter mais informações sobre os formatos de ARN, consulte a [Referência geral da Amazon Web Services](#).

- Método HTTP (string): o método a ser amostrado; por exemplo, **GET**.
- URL path (Caminho do URL) (string) — O caminho URL da solicitação.
- (opcional) Atributos (chave e valor): cabeçalhos da solicitação HTTP original; por exemplo, **Connection**, **Content-Length** ou **Content-Type**. Cada valor de atributo pode ter até 32 caracteres.

## Exemplos de regras de amostragem do X-Ray

### Exemplo de regra de amostragem Nº #1

Essa regra amostra todas as solicitações GET para a API testxray no estágio test.

- Rule name (Nome da regra — **test-sampling**
- Prioridade — **17**
- Tamanho do reservatório — **10**
- Taxa fixa — **10**
- Nome do serviço — **testxray/test**
- Tipo de serviço — **AWS::ApiGateway::Stage**
- Método HTTP — **GET**
- ARN do recurso — **\***
- Host — **\***

### Exemplo de regra de amostragem Nº #2

Essa regra amostra todas as solicitações para a API testxray no estágio prod.

- Rule name (Nome da regra — **prod-sampling**
- Prioridade — **478**
- Tamanho do reservatório — **1**
- Taxa fixa — **60**
- Nome do serviço — **testxray/prod**
- Tipo de serviço — **AWS::ApiGateway::Stage**
- Método HTTP — **\***
- ARN do recurso — **\***
- Host — **\***
- Atributos — **{}**

## Rastreamentos do AWS X-Ray para APIs do Amazon API Gateway

Esta seção discute segmentos e subsegmentos de rastreamento do AWS X-Ray, bem como outros campos de rastreamento para APIs do Amazon API Gateway.

Antes de ler esta seção, reveja os tópicos a seguir no Guia do desenvolvedor do X-Ray:

- [Use an Console de gerenciamento da AWS](#)
- [X-Ray segment documents](#)
- [Conceitos do](#)

## Tópicos

- [Exemplos de objetos de rastreamento para uma API do API Gateway](#)
- [Noções básicas sobre o rastreamento](#)

## Exemplos de objetos de rastreamento para uma API do API Gateway

Esta seção discute alguns dos objetos que você pode ver em um rastreamento para uma API do API Gateway.

## Anotações

As anotações podem aparecer em segmentos e subsegmentos. Elas são usadas como expressões de filtragem em regras de amostragem para filtrar rastreamentos. Consulte mais informações em [Configure sampling rules](#).

Veja a seguir um exemplo de um objeto annotations, em que um estágio da API é identificado pelo ID da API e o nome de estágio da API:

```
"annotations": {
 "aws:api_id": "a1b2c3d4e5",
 "aws:api_stage": "dev"
}
```

Consulte mais informações sobre anotações em [X-Ray segment documents](#) e selecione X-Ray segment documents, Annotations.

## AWS Dados de recursos da

O objeto aws aparece somente em segmentos. Veja a seguir um exemplo de um objeto aws que corresponde à regra de amostragem Padrão. Consulte uma explicação detalhada sobre as regras de amostragem em [Configure sampling rules](#).

```
"aws": {
```



```
"xray": {
 "sampling_rule_name": "Default"
},
"api_gateway": {
 "account_id": "123412341234",
 "rest_api_id": "a1b2c3d4e5",
 "stage": "dev",
 "request_id": "a1b2c3d4-a1b2-a1b2-a1b2-a1b2c3d4e5f6"
}
}
```

Consulte mais informações sobre o objeto `aws` em [X-Ray segment documents](#) e selecione X-Ray segment documents, AWS resource data.

## Noções básicas sobre o rastreamento

Veja a seguir um segmento de rastreamento para um estágio do API Gateway. Consulte uma explicação detalhada sobre os campos que compõem o segmento de rastreamento em [X-Ray segment documents](#).

```
{
 "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"
}
```

# Portais do API Gateway

É possível utilizar um portal do API Gateway para compartilhar suas APIs e a respectiva documentação com consumidores de API em uma página da Web. Um portal contém um conjunto de produtos do portal. Cada produto do portal é um agrupamento lógico de APIs REST e contém a documentação que você cria e publica para seus consumidores de API. As páginas do produto em um portal contêm a documentação personalizada em nível de produto do portal. As páginas de endpoint REST do produto contêm a documentação de cada uma das APIs REST, bem como o caminho, o método e o estágio de uma API REST. A combinação de páginas de produtos e páginas de endpoint REST de produtos fornece a documentação completa para permitir que os consumidores de API aprendam como começar a utilizar suas APIs REST.

Utilize um portal para fazer o seguinte:

- Use o API Gateway em todo o ciclo de vida das APIs, desde a criação até a documentação e a distribuição delas.
- Compartilhe produtos do portal em todas contas da AWS.
- Personalize e compartilhe um local central para seus produtos do portal e forneça páginas de produtos.

## Considerações

As considerações a seguir podem afetar o uso de portais do API Gateway:

- Você só pode criar portais para compartilhar APIs REST.
- Os portais são criados em nível de região da AWS. Seu portal só pode conter APIs REST que estejam na mesma região onde você cria o portal.
- Você só pode usar o Console de gerenciamento da AWS, a AWS CLI ou AWS SDKs para criar e compartilhar seus portais.
- Você só pode controlar o acesso ao seu portal utilizando grupos de usuários do Amazon Cognito. Os consumidores do seu portal podem fazer login diretamente por meio de um grupo de usuários ou federar por meio de um provedor de identidades (IdP) de terceiros. Se você usa um grupo de usuários do Amazon Cognito, não pode definir um segredo de cliente para ele. Para acessar mais informações, consulte [Grupos de usuários do Amazon Cognito](#).

## Produtos de portal no API Gateway

Um produto de portal representa um serviço ou funcionalidade que você deseja compartilhar. Seu produto de portal consiste em endpoints REST e páginas do produto. Os endpoints REST do produto são os pontos de acesso ao seu produto do portal e consistem no caminho e no método de uma API REST e no estágio em que ela é implantada. As páginas do produto são a documentação que você fornece para explicar como os consumidores da API podem usar os endpoints do seu produto. Um produto do portal pode conter todo o estágio Prod como uma coleção de endpoints REST do produto ou apenas o recurso GET /pets implantado no estágio Prod como um único endpoint REST do produto.

Seu produto do portal é personalizável. É possível adicionar documentação personalizada, renomear os endpoints REST do produto, reorganizar a ordem de exibição, adicionar novas seções e compartilhar produtos entre contas da AWS. Para que todas as alterações feitas em seu produto do portal sejam implementadas, você deve publicar novamente os portais que utilizem seu produto do portal.

### Exemplo de produto do portal de adoção de animais de estimação

Por exemplo, você pode ter várias APIs REST que representem um serviço de adoção de animais de estimação. É possível usar o API Gateway para criar um produto do portal pet adoption. Esse produto do portal ajudaria os clientes a descobrir quais APIs utilizar para conhecer e adotar animais de estimação. Esse produto do portal utiliza APIs REST que você já criou, mas permite reagrupá-las e organizá-las. Você também pode fornecer documentação sobre os termos e condições de uso do produto do portal de adoção de animais de estimação e permitir que os clientes experimentem suas APIs. Essas informações são todas armazenadas em seu produto do portal.

A tabela a seguir mostra três APIs que representam um produto do portal de adoção de animais de estimação e seus nomes de operação de endpoint REST e nomes de seção de página correspondentes.

| ID da API REST | Caminho e método da API REST | Estágio da API REST | Nome de operação | Seções da página |
|----------------|------------------------------|---------------------|------------------|------------------|
| kf5387miad     | GET /dogs                    | Prod                | View dogs        | AdoptAnim<br>als |

| ID da API REST | Caminho e método da API REST    | Estágio da API REST | Nome de operação | Seções da página |
|----------------|---------------------------------|---------------------|------------------|------------------|
| kf5387miad     | GET /dogs/{dogId}               | Prod                | View dog         | AdoptAnim<br>als |
| ra8obxcevg     | GET /cats                       | Prod                | View cats        | AdoptAnim<br>als |
| ra8obxcevg     | GET /cats/{catId}               | Prod                | View cat         | AdoptAnim<br>als |
| h0rpx9cm62     | ANY /user/{userId}/<br>{petId+} | Beta                | Request<br>visit | AdoptProc<br>ess |

Neste exemplo, duas APIs REST, ra8obxcevg e kf5387miad, estão agrupadas na seção AdoptAnimals. O resultado dessa navegação de ordenação seria semelhante ao seguinte em um portal:

**Pet adoption portal**

Search

**pet adoption**

**Documentation**  
Pet adoption

**API reference**

- ▼ **AdoptAnimals**
  - View cat (GET)
  - View cats (GET)
  - View dogs (GET)
  - View dog (GET)
- ▼ **AdoptProcess**
  - Request visit (GET)
  - Request visit (POST)
  - Request visit (PUT)
  - Request visit (PATCH)
  - Request visit (DELETE)
  - Request visit (OPTIONS)
  - Request visit (HEAD)

**View cat**

GET `https://ra8obxcvg.execute-api.us-west-1.amazonaws.com/Prod/cats/{catId}`

View a cat for adoption.

**Arguments**

- catId (string): The cat ID.  
Required: Yes

**Request body**

The request does not have a request body.

**Responses**

200

```
1 {
2 "catId": "string",
3 "catAge": number
4 }
```

**View cats**

GET `https://ra8obxcvg.execute-api.us-west-1.amazonaws.com/Prod/cats`

View all the cats available for adoption.

**Request body**

The request does not have a request body.

Last update published on: November 17, 2025, 20:59 (UTC-08:00)

Como o método ANY é abrangente, o portal mostra todos os métodos HTTP aceitos. Esse portal também contém documentação que foi criada pelo proprietário do produto do portal.

## Próximas etapas

Para começar a usar os produtos do portal, faça o seguinte:

- Para criar um produto do portal, consulte [the section called “Criar um produto do portal”](#).
- Para saber mais sobre o recurso Experimente, consulte [the section called “Habilitar o botão Experimente para um endpoint REST do produto em seu portal”](#).
- Para saber mais sobre páginas de produto, consulte [the section called “Criar uma página de produto”](#).
- Para saber mais sobre como compartilhar seu produto do portal, consulte [the section called “Compartilhar produtos do portal”](#).

Depois de criar um produto do portal, você pode publicá-lo em um portal. Para obter mais informações, consulte [the section called “Criar um portal”](#).

## Criar um produto do portal no API Gateway

O procedimento a seguir mostra como criar um produto do portal. Portal é uma coleção de produtos do portal. Depois de criar seu produto do portal, você vai criar endpoints REST e as páginas do produto. Para saber mais sobre como produtos do portal, consulte [the section called “Produto do portal”](#).

### Considerações

As considerações a seguir podem afetar a maneira como você cria um produto do portal:

- Seu produto do portal pode conter APIs REST privadas e públicas. As APIs privadas não comportam o recurso Experimente e, como resultado, têm uma diferença visual em seu portal. Como proprietário do portal, talvez você precise fornecer uma documentação para explicar isso.
- Se você criar seu produto do portal usando a AWS CLI ou AWS SDKs, seu portal não terá nenhum endpoint nem páginas do produto. Você precisa adicionar esses recursos utilizando a AWS CLI ou o console. Para saber como criar um endpoint REST do produto, consulte [the section called “Criar um endpoint REST de produto”](#). Para saber como criar uma função para um serviço, consulte [the section called “Criar uma página de produto”](#).

### Criar um produto do portal

O procedimento a seguir mostra como criar um produto do portal.

Como criar um produto do portal

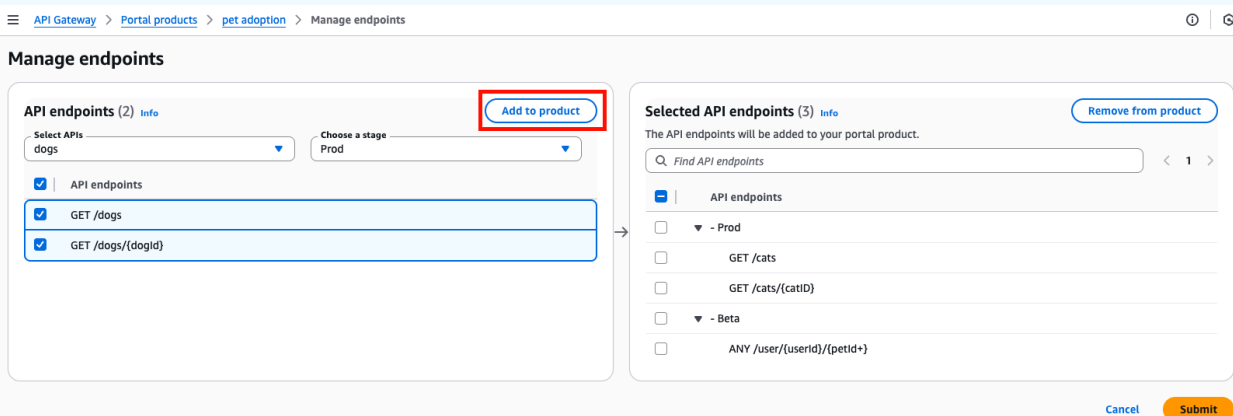
1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, selecione Produtos do portal.
3. Escolha Criar produto.
4. Em Nome do projeto, digite o nome de seu produto do portal.
5. Em Descrição do produto, insira uma descrição.
6. Escolha Next (Próximo).
7. Para selecionar os endpoints REST do seu produto, em Endpoints de API, escolha uma API e, depois, selecione um estágio.



- Para adicionar um endpoint aos endpoints REST do produto, selecione o endpoint de API e escolha Adicionar ao produto.

### Note

Não escolha Próximo sem primeiro selecionar Adicionar ao produto.



O endpoint de API aparecerá na lista Endpoints de API selecionados.

- Escolha Next (Próximo).
- Analise sua seleção e escolha Criar produto.

Depois de criar seu produto do portal usando o console, todas as suas páginas do produto e páginas de endpoint REST estarão no modo de rascunho e não aparecerão no portal. Para que as páginas do produto e as páginas de endpoint REST fiquem visíveis para os consumidores, você precisa adicionar seu rascunho a uma seção. Se você criar o produto do portal utilizando a AWS CLI ou AWS SDKs, precisará adicionar o rascunho à seção no comando AWS CLI. Independentemente de como você adiciona seu rascunho a uma seção, precisará publicar o portal que usa seus produtos para que ele fique visível para os consumidores.

Como adicionar seu rascunho a uma seção da página

- Seus rascunhos estão listados na guia Documentação. Há páginas de documentação de rascunho para as páginas do produto e páginas de referência de API de rascunho para as páginas de endpoint REST do produto. Escolha as páginas de referência de API de rascunho.
- Escolha a página de referência de API de rascunho.

Se você não tiver nenhum endpoint REST de produto, não terá nenhuma página de referência de API de rascunho. Para saber como criar um endpoint REST do produto, consulte [the section called “Criar um endpoint REST de produto”](#).

3. Selecione Editar página.
4. Nessa página, você pode substituir qualquer parte existente da documentação da API ou usar a documentação do API Gateway. Para permitir que o conteúdo da página de endpoint REST do seu produto fique visível para os consumidores, em Nome da seção, insira um nome. Se esse fosse o endpoint `/pets-GET`, o nome da página poderia ser `Pets`.
5. Escolha Salvar alterações.
6. O novo nome da página que você criou aparece na seção Páginas de referência de API.

Para permitir que a nova página fique visível para os consumidores, você ainda precisa publicar novamente o portal. Para obter mais informações, consulte [the section called “Publicar um portal”](#).

## Criar um endpoint REST de produto no API Gateway

Um endpoint REST do produto é um ponto de acesso ao seu produto do portal. Cada endpoint consiste no caminho e no método de uma API REST e no estágio em que ela é implantada. O endpoint inclui qualquer documentação que você definiu para sua API REST ou pode ser atualizado com documentação personalizada. Ao criar o endpoint REST do produto, você define as seguintes configurações:

- O nome da operação do endpoint REST do produto. Se sua API REST tiver o caminho e o método `GET /pets`, em um portal, o nome será `/pets`. Usando o nome da operação, você pode alterar o nome do endpoint REST do produto para `Pet summaries`. Isso não afeta sua API existente.
- O recurso Experimente para o endpoint do seu produto. Isso permite que os consumidores de API experimentem as APIs em seu portal. Se você criar um endpoint de produto para uma API privada, o recurso Experimente será definido como `UNAVAILABLE`. Para obter mais informações, consulte [the section called “Habilitar o botão Experimente para um endpoint REST do produto em seu portal”](#).
- Se você atualmente tem partes de documentação definidas para sua API REST, poderá importar a documentação ou substituir a existente e criar outra em nível de produto utilizando o [GitHub Flavored Markdown](#). A substituição de documentação não é propagada de volta para as partes da

documentação da API do API Gateway. Se você importar suas partes da documentação atual, o API Gateway sincronizará as alterações em sua API com o endpoint REST do produto.

## Criar um endpoint REST de produto

O procedimento a seguir mostra como adicionar um endpoint REST de um produto a um produto do portal existente. Para saber como criar um produto do portal e um endpoint REST do produto juntos, consulte [the section called “Criar um produto do portal”](#).

Como criar um endpoint REST de produto para um portal de produtos

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, selecione Produtos do portal.
3. Escolha um produto do portal.
4. Escolha a guia Endpoints associados e, depois, Adicionar endpoints.
5. Para selecionar os endpoints REST do seu produto, em Endpoints de API, escolha uma API e, depois, selecione um estágio.
6. Para adicionar um endpoint aos endpoints REST do produto, selecione o endpoint de API e escolha Adicionar ao produto.

O endpoint de API aparecerá na lista Endpoints de API selecionados.

7. Selecione Enviar.
8. A página de endpoint REST do produto agora é um rascunho. Escolha Páginas de referência de API de rascunho e, depois, escolha a página do endpoint REST do produto para finalizar o conteúdo da página.
9. Selecione Editar página.
10. Para substituir a documentação existente do API Gateway, em Fonte de documentação, ative Substituir a documentação e faça o seguinte:
  - a. Em Exibir conteúdo, escolha Criar substituição.
  - b. Em Nome da operação, insira um novo nome de operação.
  - c. Em Corpo da página, insira sua documentação personalizada.
  - d. (Opcional) Em Endpoint, insira um endpoint personalizado que aparece no seu portal.

11. Para permitir que os clientes invoquem sua API no portal, selecione Recurso Experimental. Para obter mais informações, consulte [the section called “Habilitar o botão Experimente para um endpoint REST do produto em seu portal”](#).
12. Em Nome da seção, insira um nome ou escolha uma seção existente. Quando você adiciona o endpoint REST do seu produto a uma seção, os consumidores podem visualizar seu conteúdo em um portal publicado.
13. Escolha Salvar alterações.

Para permitir que a nova página fique visível para os consumidores, você ainda precisa publicar novamente o portal. Para obter mais informações, consulte [the section called “Publicar um portal”](#).

## Criar uma página do produto no API Gateway

Uma página de produto ajuda seus consumidores a entender e usar seus produtos. A documentação está em nível de produto, portanto, todos os portais que incluem seu produto têm essa documentação. O API Gateway comporta dois tipos de páginas de produtos escritas no [GitHub Flavored Markdown](#):

### Página Visão geral

Quando você cria seu produto, o API Gateway gera automaticamente uma página Visão geral para seu produto com base nas informações que você fornece ao criar seu produto. Essa página tem uma formatação adicional que não pode ser alterada. Você pode modificar o texto ou remover a página do produto do portal.

### Páginas personalizadas

Página personalizada é uma página escrita em Markdown. Você pode usar essa seção para ver os termos e condições ou um tutorial sobre como usar seu produto. Você pode transformar qualquer página personalizada em uma página de visão geral. Quando você transforma uma página personalizada em uma página de visão geral, o conteúdo da página é formatado para corresponder à página de visão geral.

## Considerações

As considerações a seguir podem afetar o uso de páginas do produto:

- Essa documentação não está em nível de API e não inclui métodos, recursos ou parâmetros de solicitação. Para saber como modificar a documentação em nível de API de um produto, consulte [the section called “Atualizar um endpoint REST de produto”](#).
- Você não pode fazer upload diretamente de um arquivo .md para o seu produto.
- É possível modificar a ordem das páginas do produto modificando a lista de páginas na ordem de exibição do seu produto do portal. Para obter mais informações, consulte [the section called “Atualizar um produto do portal”](#).
- Você precisa publicar novamente todos os portais que incluam seu produto para que as alterações na documentação sejam implementadas.
- Se você tiver vários portais que usam o mesmo produto, as páginas do produto serão as mesmas em todos os portais. Para ter os mesmos endpoints REST do produto, mas páginas de produto diferentes, crie vários produtos.

## Criar uma página de produto

O procedimento a seguir mostra como criar um produto do produto.

Como criar uma página do produto

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, selecione Produtos do portal.
3. Escolha um produto.
4. Em Documentação, escolha Criar página personalizada.
5. Em Título da página, insira o título da página.
6. Em Corpo, insira sua documentação complementar usando o GitHub Flavored Markdown.

A seção Versão prévia mostra como seu conteúdo aparece em um portal. O estilo visual final pode mudar com base nas configurações do seu portal.

7. Escolha Criar página do produto.
8. Sua página do produto agora está no modo de rascunho. Escolha Páginas de documentação de rascunho e, depois, escolha a página do produto para finalizar o conteúdo dela.
9. Selecione Editar página.
10. Em Nome da seção, insira um nome ou escolha uma seção existente. Quando você adiciona a página a uma seção, os consumidores podem visualizar seu conteúdo em um portal publicado.

## 11. Escolha Salvar alterações.

Para permitir que a nova página fique visível para os consumidores, você ainda precisa publicar novamente o portal. Para obter mais informações, consulte [the section called “Publicar um portal”](#).

## Atualizar um produto do portal no API Gateway

Ao atualizar um produto do portal, você pode alterar o nome, a descrição ou a ordem de exibição de como os endpoints REST e as páginas do produto aparecem. Para modificar a ordem de exibição, você modifica a lista de páginas das páginas do produto e o nome da seção dos endpoints do produto. Se você fizer isso usando a AWS CLI ou os SDKs, especifique o ARN da página do produto ou o ARN do endpoint REST do produto. Você precisa publicar novamente todos os portais que usam seus produtos para que seus consumidores de API vejam as alterações exibidas em um portal.

### Atualizar um produto do portal

O procedimento a seguir mostra como alterar a lista de seções para reordenar os endpoints REST do produto conforme eles aparecerão em um portal. Neste procedimento, presumimos que você tenha pelo menos dois endpoints REST em seu produto do portal.

Como atualizar um produto do portal

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, selecione Produtos do portal.
3. Escolha um produto.
4. Na guia Documentação, em Páginas do produto, escolha Gerenciar.
5. Arraste e solte os endpoints REST do produto para reordená-los. Você também pode mover os endpoints REST e as páginas do produto para a seção de páginas de rascunho. As páginas nas páginas de rascunho não ficarão visíveis no seu portal.
6. (Opcional) Troque os nomes dos endpoints REST do produto ou adicione uma nova seção de referência de API. Essas alterações não afetarão suas APIs REST.
7. Escolha Salvar alterações.

## Atualizar um endpoint REST de um produto no API Gateway

Ao atualizar um endpoint REST do produto, você pode alterar todas as configurações do endpoint do produto, exceto o caminho e o método de uma API REST e o estágio em que ela é implantada.

Todas as alterações nos endpoints do seu produto são atualizadas automaticamente, mas você precisa publicar novamente todos os portais que usam seus produtos para ver as alterações exibidas no portal.

## Atualizar um endpoint REST de produto

O procedimento a seguir mostra como atualizar um endpoint REST do produto para substituir a documentação atual da API pela documentação de referência de API personalizada.

Como atualizar um endpoint REST de produto

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, selecione Produtos do portal.
3. Escolha um produto.
4. Na guia Documentação, em Páginas de referência de API, escolha o nome de um endpoint REST do produto, como /dogs - GET.
5. Em Versão prévia, escolha Editar página.
6. Em Fonte de documentação, ative Substituir a documentação.
7. Em Exibir conteúdo, escolha Substituir o conteúdo existente.

Se você escolher Remover todo o conteúdo, o conteúdo será removido da página, mas você poderá escolher Substituir o conteúdo existente para acessar a documentação novamente.

8. Em Corpo da página, insira a documentação de referência de API personalizada utilizando o [GitHub Flavored Markdown](#).

Essa documentação não será propagada de volta para o API Gateway.

9. Escolha Salvar alterações.
10. Você precisa publicar novamente todos os portais que usam o produto em questão para as alterações serem implementadas.

## Atualizar uma página do produto no API Gateway

Quando você atualizar uma página de produto, poderá alterar todas as configurações da página do produto. Todas as alterações na página do seu produto são atualizadas automaticamente, mas você precisa publicar novamente todos os portais que usam seus produtos para ver as alterações exibidas no portal.

## Compartilhar produtos do portal no API Gateway

Você pode compartilhar um produto do portal em todas as Contas da AWS utilizando o AWS RAM. Quando você compartilha um produto do portal, a outra conta pode utilizá-lo no próprio portal. Com produtos do portal compartilhados, você pode criar um único catálogo das APIs da sua organização e aplicar padrões de governança em todo o seu ecossistema de APIs. Ao mesmo tempo, o compartilhamento de produtos do portal oferece flexibilidade para os provedores de API desenvolverem, testarem e manterem APIs em suas próprias contas.

### Considerações

As considerações a seguir podem afetar a maneira como você compartilha recursos do portal:

- Quando você compartilha seu produto com outra conta, esta não pode modificar nenhuma propriedade da sua API REST. Isso inclui os endpoints de integração, a estratégia de autorização ou a configuração do estágio.
- Quando você adiciona o produto do portal de outra conta ao seu portal, o proprietário do produto não pode visualizar nem controlar nenhuma outra propriedade do seu portal. O proprietário do produto do portal apenas sabe que o produto está sendo usado no seu portal.
- Os produtos do portal do API Gateway são compartilhados em nível de Região da AWS.
- É possível usar um compartilhamento de recursos com várias entidades principais e, depois de criar o compartilhamento de recursos, adicionar mais entidades principais a ele. Recomendamos que, quando possível, você reutilize seu compartilhamento de recursos.
- Se as duas contas estiverem na mesma organização que está utilizando o AWS Organizations, o compartilhamento de recursos será aceito automaticamente. Ainda assim você precisa criar o compartilhamento de recursos usando o AWS RAM.
- Se as duas contas estiverem na mesma organização que está usando o AWS Organizations e o compartilhamento de recursos dentro da organização estiver habilitado, todas as entidades principais na organização com as quais você compartilhar receberão acesso automaticamente aos compartilhamentos de recursos. Não há necessidade de convite, e você pode ignorar o compartilhamento de recursos.
- Se a conta com a qual você compartilhou o produto não aceitar o compartilhamento de recursos em 12 horas, você deverá compartilhar o recurso novamente.
- Após a criação do compartilhamento de recursos, o AWS RAM atualiza a política de compartilhamento referente ao produto a fim de impedir o acesso a entidades principais sem o



acesso explícito `allow`. Para ter mais informações, consulte [Determining whether a request is allowed or denied within an account](#) no Guia do usuário do IAM.

A política de recurso atualizada será semelhante ao seguinte:

```
{
 "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/*"
]
 }
]
}
```

O AWS RAM impediu que entidades principais sem acesso “allow” explícito adicionassem o produto ao portal, adicionando o seguinte:

```
"StringNotEquals": { "aws:PrincipalAccount": "555555555555" }
```

Para saber como compartilhar um produto, consulte [the section called “Compartilhar seu produto do portal com um proprietário do portal”](#). Para saber como adicionar um produto compartilhado com você ao seu portal, consulte [the section called “Adicionar um produto compartilhado ao seu portal”](#).

## Compartilhar seu produto do portal com um proprietário do portal no API Gateway

Como proprietário do produto, você pode compartilhá-lo com um proprietário do portal em outra Conta da AWS utilizando AWS RAM. Você mantém o controle total de todas as páginas do produto e das páginas de endpoint REST do produto.

### Considerações

As seguintes considerações podem afetar a maneira como você compartilha produtos do portal:

- Você pode compartilhar ou cancelar o compartilhamento de seus produtos do portal a qualquer momento. Se você cancelar o compartilhamento de um produto do portal enquanto ele estiver sendo utilizado em um portal publicado, o proprietário do portal não poderá visualizar o produto, visualizar nenhuma atualização feita nele nem publicar novamente ou visualizar o portal até remover o produto.
- Você pode visualizar os produtos do portal compartilhados com outras contas.
- Você não pode modificar nenhum portal que contenha seu produto do portal, a menos que você crie o próprio portal.

### Compartilhar o produto do portal

O procedimento a seguir mostra como criar um compartilhamento de recursos.

### Console de gerenciamento da AWS

Para usar o Console de gerenciamento da AWS, consulte [Creating a resource share in AWS RAM](#) no Guia do usuário do AWS RAM.

Em Selecionar tipo de recurso, escolha Produto do API Gateway.

### AWS CLI

O [create-resource-share](#) indicado abaixo cria um compartilhamento de recursos para seu nome de domínio personalizado privado. Pode levar alguns minutos para que as associações de entidades principais entre recurso e principal sejam concluídas. Para entidades principais, forneça um ID de conta ou um ID de organização, como `arn:aws:organizations::123456789012:organization/o-1234abcd`. Você pode fornecer várias entidades principais para seu compartilhamento de recursos.

```
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 cancelar o compartilhamento do seu produto do portal, utilize AWS RAM para excluir o compartilhamento de recursos.

A qualquer momento, você pode modificar a política de compartilhamento de produtos a fim de modificar quais entidades principais podem utilizar seus produtos nos respectivos portais.

Como atualizar a política de compartilhamento de produtos

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, selecione Produtos do portal.
3. Escolha um produto.
4. Na guia Compartilhamento de produtos, em Política de compartilhamento de produtos, selecione Gerenciar política de compartilhamento de produtos.
5. Se você não compartilhou seu produto antes, sua política de compartilhamento de produtos negará todo o acesso a outras contas dos seus produtos do portal. Você precisa atualizar a política para permitir o acesso de determinadas contas ao seu produto do portal.
6. Depois de atualizar sua política de compartilhamento de produtos, escolha Salvar alterações.

## Adicionar um produto compartilhado ao seu portal no API Gateway

Como proprietário de um portal, você pode utilizar produtos do portal compartilhados com você por outras contas no seu portal. Você mantém total controle do seu portal.

### Considerações

As considerações a seguir podem afetar a maneira como você compartilha recursos do portal:

- Você precisa aceitar o compartilhamento de recursos do produto do proprietário para utilizar o produto dele em seu portal. Se você e o proprietário do produto estiverem nas mesmas organizações, o AWS RAM poderá concluir algumas etapas de compartilhamento para você.

- Se o proprietário do produto cancelar o compartilhamento de um produto enquanto ele estiver no seu portal publicado, ele ainda estará visível no portal, mas você não poderá publicar o portal novamente até removê-lo. É sua responsabilidade, como proprietário do portal, remover o produto, pois o proprietário não pode fazer isso por você.
- Você pode visualizar os produtos do portal, as respectivas páginas e as páginas do endpoint REST do produto, mas não pode modificar nenhum desses recursos.
- Se um produto for compartilhado, você não poderá compartilhá-lo com outra conta.

### (Opcional) Aceitar o compartilhamento de recursos

Se o proprietário do produto criar um compartilhamento de recursos, você terá 12 horas para aceitá-lo. Se você estiver na mesma organização que está usando o AWS Organizations como proprietário do produto, o compartilhamento será aceito automaticamente. Se estiver em uma organização que tem recursos compartilhados automáticos habilitados, o recurso será compartilhado automaticamente com você.

### Console de gerenciamento da AWS

Para usar o Console de gerenciamento da AWS, consulte [Accepting and rejecting resource share invitations](#) no Guia do usuário do AWS RAM.

### AWS CLI

Para encontrar todos os recursos compartilhados com você, use o seguinte comando [get-resource-share-invitations](#):

```
aws ram get-resource-share-invitations \
 --region us-west-2
```

Use o ARN de compartilhamento de recursos resultante para aceitar o convite de compartilhamento de recursos. O comando [accept-resource-share-invitation](#) a seguir aceita o compartilhamento de recursos.

```
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
```

## Adicionar um produto compartilhado ao seu portal

Depois de aceitar o compartilhamento de recursos, você vai adicionar o produto ao seu portal.

### Console de gerenciamento da AWS

Como adicionar um produto compartilhado ao seu portal

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, selecione Portais.
3. Escolha um portal.
4. Na guia Produtos, para Produtos do portal, escolha Adicionar produtos.
5. Adicione um produto ao seu portal. Os produtos compartilhados são exibidos como shared na lista de produtos.
6. Escolha Salvar alterações.

### AWS CLI

Para localizar todos os produtos do portal compartilhados com você, utilize o seguinte comando `get-portal-products`:

```
aws apigatewayv2 get-portal-products \
 --resource-owner OTHER_ACCOUNTS \
 --region us-west-2
```

Para adicionar um produto compartilhado ao seu portal, utilize o seguinte comando `update-portal`:

```
aws apigateway update-portal \
 --included-portal-product-arns arn:aws:apigateway:us-west-2:111122223333:/portalproducts/p0000000000 \
 --region us-west-2
```

## Habilitar o botão Experimente para um endpoint REST do produto do API Gateway em seu portal

Use o botão Experimente para permitir que um consumidor de API invoque o endpoint do seu produto por meio do seu portal. Quando um consumidor de API usa o botão Experimente, ele insere os parâmetros de solicitação do método e invoca o endpoint do produto. Depois, o API Gateway invoca sua API pela Internet pública e exibe a resposta na janela Experimente. Você pode usar uma página de produto personalizada para fornecer informações adicionais sobre os parâmetros necessários para acessar sua API.

O API Gateway utiliza os seguintes limites para proteger suas APIs:

- O API Gateway permite apenas três solicitações por segundo para sua API.
- O API Gateway utiliza um tempo limite integrado de 29 mil ms. Sua API real pode ter um limite de tempo maior, mas o API Gateway não aplica esse tempo limite quando um cliente utiliza o botão Experimente.
- O API Gateway limita a carga útil de resposta a 6 MB.

### Considerações

As seguintes considerações podem afetar a maneira como você realiza um teste:

- O botão Experimente não é aceito quando você visualiza um portal.
- O botão Experimente não é aceito por APIs REST com os seguintes recursos:
  - APIs privadas
  - APIs que usam TLS mútuo.
  - APIs que usam certificados SSL privados ou autoassinados.

Como proprietário do portal, você é responsável por se comunicar com seus consumidores de API. Esse é o motivo pelo qual o botão Experimente não existe para nenhuma API REST que não seja aceita. O API Gateway não explica isso para você.

## Habilitar o botão Experimente para um endpoint REST do produto

O procedimento a seguir mostra como ativar o botão Experimente para um endpoint do produto.

Para saber como utilizar o botão Experimente em um portal, consulte [the section called “Usar um portal”](#).

Habilitar o botão Experimente para um endpoint REST do produto

1. No painel de navegação principal, selecione Produtos do portal.
2. Escolha um produto.
3. Na guia Documentação, em Páginas de referência de API, escolha o nome de um endpoint REST do produto, como /dogs - GET.
4. Selecione Editar página.
5. Para permitir que os clientes invoquem sua API no portal, selecione Recurso Experimente.
6. Escolha Salvar alterações.
7. Você precisa publicar novamente todos os portais que usem o produto em questão para as alterações serem implementadas.

## Excluir um produto do portal no API Gateway

Quando você excluir um produto do portal, não será possível recuperá-lo.

### Excluir um produto do portal

O procedimento a seguir mostra como excluir um produto do portal.

Como excluir um produto do portal

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, selecione Produtos do portal.
3. Escolha um produto do portal.
4. Escolha Excluir.
5. Confirme sua escolha e selecione Excluir.
6. Você precisa publicar novamente todos os portais que usem o produto em questão para as alterações serem implementadas.

# Criar um portal no API Gateway

Portal é uma coleção de produtos. Você controla a autorização, a identidade visual e a publicação do seu portal. Quando você cria um portal, seu status de publicação é `Disabled`. Isso significa que seu portal não pode ser descoberto na Web. Para os consumidores acessarem seu portal, você precisa publicá-lo. Você também pode visualizar seu portal antes de publicá-lo.

O API Gateway comporta portais nos seguintes navegadores:

- Firefox
- Google Chrome
- Microsoft Edge
- Safari

## Escolher um nome do domínio

Por padrão, o URL do seu portal é hospedado no seguinte nome de domínio de propriedade da Amazon:

```
https://p-portalId.apigw-portal.us-east-1.on.aws
```

Para personalizar o nome de domínio, você pode fornecer um de sua propriedade e um certificado SSL para o domínio personalizado. O certificado deve ser associado ao [AWS Certificate Manager](#) (ACM) na região do Leste dos EUA (Norte da Virgínia) e deve ser aceito no Amazon CloudFront. Para acessar informações, consulte [Requisitos para usar certificados SSL/TLS com o CloudFront](#).

Quando você usa um nome de domínio personalizado, o API Gateway cria uma distribuição do CloudFront, protegida em trânsito com seu certificado ACM. Depois, o API Gateway fornece um registro de alias, que você vai adicionar à sua configuração DNS. O registro de alias é o nome de domínio de propriedade da Amazon. Você deve adicioná-lo para encaminhar o tráfego para a distribuição do CloudFront para seu domínio personalizado. O nome de domínio fornecido deve ser exclusivo e ainda não ter sido usado pela lista de aliases em uma distribuição existente do CloudFront.

Se você usar seu próprio nome de domínio, o API Gateway utilizará a política de segurança TLS\_1\_2 para proteger seu portal. Para acessar mais informações, consulte [Protocolos e cifras compatíveis entre visualizadores](#). Você não pode alterar esse valor.



## Criar um portal

Você vai usar o console do API Gateway para criar e configurar seu portal. Os procedimentos a seguir guiam você por todas as etapas necessárias. Você também pode escolher Ir para revisar e criar no final do primeiro procedimento para criar um portal vazio.

O procedimento a seguir mostra como criar um portal. Primeiro, defina os detalhes do portal.

Como definir os detalhes do seu portal

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, selecione Portais.
3. Escolha Criar portal.
4. Em detalhes do portal, em Nome da política, insira um nome.
5. Em Descrição do portal, insira uma descrição.
6. Em Autorização do portal, selecione como autorizar seu portal.
  - Se essa for a primeira vez que cria um portal, recomendamos que você escolha Nenhum. Se você planeja adicionar quaisquer APIs de produção ao seu portal, utilize Exigir autorização.
  - Para exigir que os consumidores do portal se autentiquem para usar seu portal, escolha Exigir autorização e faça o seguinte:
    - a. Para o grupo de usuários do Amazon Cognito, selecione seu grupo de usuários.

O grupo de usuários deve estar na mesma região que seu portal.
    - b. Em ID do cliente da aplicação, escolha o ID do cliente da aplicação.

No seu cliente de aplicação, você deve definir o URL de retorno de chamada para o URL padrão do portal.
    - c. Em Domínio do grupo de usuários, insira o domínio do grupo de usuários. Para acessar mais informações, consulte [Configurar um domínio do grupo de usuários](#).
7. Em Configuração de domínio, selecione o nome de domínio para seu portal.
  - Se essa for a primeira vez que cria um portal, recomendamos que você escolha Domínio padrão. Depois de publicar seu portal, ele estará disponível em `https://p-portalId.apigw-portal.us-east-1.on.aws`.

- Para fornecer um nome de domínio personalizado de sua propriedade, selecione Nome de domínio personalizado e faça o seguinte:
  - a. Em Nome do domínio, insira o nome de domínio totalmente qualificado.
  - b. Em Certificado ACM, escolha um certificado ACM que abranja o nome do domínio.

Você precisará adicionar o registro de alias que o API Gateway provisiona no seu registro DNS, a fim de direcionar o tráfego do seu portal.

8. Para adicionar um monitor de aplicação CloudWatch RUM ao seu portal, em Monitor de aplicação RUM, escolha um monitor de aplicação. Para acessar mais informações, consulte [CloudWatch RUM](#).
9. Escolha Next (Próximo).

Depois de definir os detalhes do seu portal, adicione produtos do portal. Você pode alterar os produtos em seu portal a qualquer momento, mas precisa publicá-lo novamente para que as alterações sejam implementadas.

Como selecionar seus produtos do portal

1. Na lista de recursos, selecione os produtos a serem adicionados ao seu portal.
2. Escolha Next (Próximo).

Depois de selecionar seus produtos, personalize o design do portal. Para fazer upload de um logotipo para o portal, você precisa ter uma imagem carregada no Amazon S3. Você pode alterar esses valores a qualquer momento, mas precisa publicar novamente o portal para que as alterações sejam implementadas.

(Opcional) Como personalizar seu portal

1. Em Logotipo do portal, forneça o URI do S3 para seu arquivo.

Somente .svg, .png ou .jpg são aceitos.

2. Para personalizar o design do portal, você tem duas opções:
  - a. Você pode escolher o tema do portal para aplicar um tema de cores predefinido ao seu portal.

- b. Você pode utilizar os seletores de cores para selecionar todas as opções do design do seu portal.
3. Escolha Next (Próximo).

Depois de selecionar o design, revise as informações do portal.

Como revisar o portal

- Revise as configurações na página de análise e criação. Quando estiver satisfeito com as configurações, clique em Criar.

Se você usar um grupo de usuários do Amazon Cognito para controlar o acesso ao portal, deverá definir o URL de retorno de chamada como o URL padrão do portal no cliente da aplicação. Para acessar mais informações, consulte [Configurações específicas da aplicação com clientes da aplicação](#).

Seu portal não estará disponível para os consumidores depois que você o criar. Você deve publicá-lo para ele ficar acessível na internet. Para obter mais informações, consulte [the section called “Publicar um portal”](#).

## Atualizar um portal no API Gateway

Você pode atualizar o tema, o logotipo ou os produtos no seu portal. Quando você atualiza seu portal, as alterações não são implementadas até você publicar novamente o portal.

### Atualizar um portal

O procedimento a seguir mostra como atualizar o design de um portal.

Como atualizar o design de um portal

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, selecione Portais.
3. Escolha um portal.
4. Na guia Design do portal, escolha Editar.
5. Para atualizar o logotipo do portal, escolha Atualizar logotipo do portal. Você pode remover o logotipo existente ou usar um novo logotipo.

6. Faça todas as atualizações de tema ou plano de fundo no design do seu portal.
7. Escolha Salvar alterações.
8. (Opcional) Escolha Gerar versão prévia do portal para ver suas alterações.

Depois de confirmar suas alterações, publique o portal.

9. Escolha Publicar portal.
10. Insira uma descrição para sua atualização do portal e escolha Publicar.

## Visualizar um portal no API Gateway

Você pode visualizar seu portal antes de publicá-lo. O API Gateway cria um URL de curta duração que você pode utilizar para visualizar a versão do seu portal que os clientes verão quando seu portal for publicado.

### Considerações

As considerações a seguir podem afetar a maneira como você visualiza um portal:

- Veja se não está habilitado para seu portal de visualização. Para obter mais informações, consulte [the section called “Habilitar o botão Experimente para um endpoint REST do produto em seu portal”](#).

### Visualizar um portal

O procedimento a seguir mostra como visualizar um portal.

Como visualizar um portal

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, selecione Portais.
3. Escolha um portal.
4. Escolha Gerar versão prévia do portal.

Não atualize sua página.

5. Depois que o API Gateway gerar sua visualização, escolha Abrir visualização do portal.

A visualização prévia do portal será aberta em uma nova guia.

# Publicar um portal no API Gateway

Para os consumidores da API acessarem seu portal, você deve publicá-lo. O URL de um portal pode ser descoberto por qualquer pessoa na Internet. Recomendamos que você visualize e proteja o portal antes de publicá-lo.

## Considerações

O API Gateway pode levar alguns minutos para publicar seu portal. Você pode monitorar o status de publicação no console.

## Publicar um portal

O procedimento a seguir mostra como publicar um portal.

Como publicar um portal

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, selecione Portais.
3. Escolha um portal.
4. Escolha Publicar portal.
5. (Opcional) Em Descrição das alterações, insira uma descrição da alteração.

Quando você publica um portal, recomendamos sempre fornecer uma breve descrição de suas alterações.

6. Selecione Publish.

O API Gateway pode levar alguns minutos para acabar de publicar seu portal. O API Gateway fornece um link para seu portal quando ele está disponível.

Para excluir seu portal, você deve primeiro desabilitá-lo. Para obter mais informações, consulte [the section called “Desabilitar um portal”](#).

## Usar um portal do API Gateway

Depois de publicar seu portal, ele estará disponível na web. Para utilizar seu portal, copie o nome de domínio padrão do portal ou o nome do domínio e insira-o em um navegador. Se o seu portal utilizar um nome de domínio, o nome de domínio padrão não poderá ser utilizado para acessar o portal.

## Usar um portal

O procedimento a seguir mostra como acessar um portal e utilizar o botão Experimente relativo a um endpoint REST do produto.

### Como usar um portal

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, selecione Portais.
3. Escolha um portal.
4. Copie o nome de domínio padrão do portal ou o nome do domínio e insira-o em um navegador.
5. Na página inicial do portal, você pode pesquisar produtos do portal e descobrir páginas de produtos. Utilize a pesquisa na parte superior do portal para pesquisar todo o conteúdo no seu portal. Utilize a barra de pesquisa em Produtos para pesquisar produtos do portal.
6. Escolha um produto do portal.
7. Escolha um endpoint REST do produto.
8. Na janela Experimente, insira os parâmetros para testar uma chamada para sua API.
9. Escolha Experimente. O API Gateway invocará sua API e mostrará os resultados na janela.

## Solucionar problemas no portal

O tópico a seguir fornece orientações para a solução de erros e problemas que você pode encontrar ao usar um portal.

### O site da página da web do portal não pode ser acessado

Se você receber uma mensagem de erro informando que o site da página do portal não pode ser acessado, verifique se publicou o portal. Se você usou um nome de domínio, verifique se adicionou o registro de alias que o API Gateway forneceu para encaminhar o tráfego para a distribuição do CloudFront para seu domínio personalizado.

### O recurso Experimente falhou

Se você não conseguir usar o recurso Experimente de um endpoint REST do produto, verifique se o estado dele é ACTIVE. Além disso, verifique se você inseriu os parâmetros corretos para invocar sua API.

## As páginas não aparecem no portal

Se você criou uma página de endpoint REST de produto ou uma página de produto e ela não está visível em seu portal, verifique se suas páginas não estão no modo de rascunho. Para verificar isso, faça o seguinte:

Como verificar se suas páginas não estão modo de rascunho

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, selecione Produtos do portal.
3. Escolha o produto do portal.
4. Na guia Documentação, confirme se suas páginas não estão na seção Páginas de rascunho de documentação ou Páginas de referência de API de rascunho, mas sim nas Páginas de documentação personalizada ou Páginas de referência de API.

## Desabilitar um portal no API Gateway

Quando você desabilita um portal, ele não está mais disponível na Web. Depois de desabilitar um portal, você pode excluí-lo ou publicá-lo novamente, se necessário.

### Desabilitar um portal

O procedimento a seguir mostra como desabilitar um portal.

Como desabilitar um portal

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, selecione Portais.
3. Escolha um portal.
4. Escolha Ações e, depois, Desabilitar portal.
5. Confirme sua escolha e escolha Desativar portal.

O status de publicação mudará para Desabilitado.

Para remover o status de desativação do portal, publique-o novamente.

# Excluir um portal no API Gateway

Quando você exclui um portal, ele não pode ser recuperado e você deve desabilitá-lo primeiro. Se desejar remover o portal da Web, você poderá desabilitá-lo. Isso permite que você modifique seu portal e publique-o novamente mais tarde.

## Excluir um portal

O procedimento a seguir mostra como excluir um portal.

Como excluir um portal

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação principal, selecione Portais.
3. Escolha um portal.
4. Se seu portal não estiver desabilitado, desabilite-o escolhendo Ações, Desabilitar portal.

Serão necessários alguns minutos para que o portal seja desabilitado. Você pode monitorar o Status da publicação para ver quando o status de publicação do seu portal está desabilitado.

5. Depois de cancelar a publicação do portal, você poderá excluí-lo. Para excluir seu portal, escolha Ações, Excluir portal. Confirme sua escolha e selecione Excluir.



# APIs HTTP do API Gateway

APIs REST e APIs HTTP são produtos da API RESTful. As APIs REST são compatíveis com mais recursos do que as APIs HTTP, enquanto as APIs HTTP são projetadas com recursos mínimos para que possam ser oferecidas por um preço mais baixo. Para ter mais informações, consulte [the section called “Escolher entre APIs REST e APIs HTTP”](#).

Você pode usar APIs HTTP para enviar solicitações para funções AWS Lambda ou para qualquer endpoint HTTP roteável. Por exemplo, você pode criar uma API HTTP que se integre a uma função do Lambda no backend. Quando um cliente chama sua API, o API Gateway envia a solicitação para a função do Lambda e retorna a resposta da função para o cliente.

APIs HTTP são compatíveis com as autorizações [OpenID Connect](#) e [OAuth 2.0](#). Eles são fornecidos com suporte incorporado para compartilhamento de recursos de origem cruzada (CORS) e implantações automáticas.

Você pode criar APIs HTTP usando o Console de Gerenciamento da AWS, a AWS CLI, APIs, CloudFormation ou SDKs.

## Tópicos

- [Desenvolver APIs HTTP no API Gateway](#)
- [Publicar APIs HTTP para os clientes invocarem](#)
- [Proteger as APIs HTTP no API Gateway](#)
- [Monitorar APIs HTTP no API Gateway](#)
- [Solução de problemas com APIs HTTP no API Gateway](#)

## Desenvolver APIs HTTP no API Gateway

Esta seção fornece detalhes sobre os recursos necessários do API Gateway durante o desenvolvimento de suas APIs do API Gateway.

Ao desenvolver a API do API Gateway, você decidirá uma série de características da API. Essas características dependem do caso de uso da sua API. Por exemplo, talvez você queira permitir que somente determinados clientes chamem sua API ou talvez queira disponibilizá-las a todos. É recomendável que uma chamada de API execute uma função do Lambda, faça uma consulta de banco de dados ou chame uma aplicação.

## Tópicos

- [Criar uma API HTTP](#)
- [Criar rotas para APIs HTTP no API Gateway](#)
- [Tipos de endereço IP para APIs HTTP no API Gateway](#)
- [Controlar e gerenciar o acesso a APIs HTTP no API Gateway](#)
- [Criar integrações para APIs HTTP no API Gateway](#)
- [Configurar CORS para APIs HTTP no API Gateway](#)
- [Transformar solicitações e respostas de API para APIs HTTP no API Gateway](#)
- [Usar definições do OpenAPI para APIs HTTP no API Gateway](#)

## Criar uma API HTTP

Para criar uma API funcional, você deve ter pelo menos uma rota, integração, estágio e implantação.

Os exemplos a seguir mostram como criar uma API com uma integração de AWS Lambda ou HTTP, uma rota e um estágio padrão configurado para implantar alterações automaticamente.

Este guia pressupõe que você já está familiarizado com o API Gateway e o Lambda. Para obter uma lista mais detalhada, consulte [Conceitos básicos](#).

## Tópicos

- [Crie uma API HTTP usando o Console de gerenciamento da AWS](#)
- [Criar uma API HTTP usando a CLI da AWS](#)

## Crie uma API HTTP usando o Console de gerenciamento da AWS

1. Abra o [console do API Gateway](#).
2. Selecione Create API (Criar API).
3. Em HTTP API (API HTTP), escolha Build (Criar).
4. Escolha Add integration (Adicionar integração) e, em seguida, escolha uma função do AWS Lambda ou insira um endpoint HTTP.
5. Em Name (Nome), insira um nome para a sua API.
6. Selecione Review and create.
7. Escolha Create (Criar).

Agora sua API está pronta para invocar. Você pode testar sua API inserindo seu URL de invocação em um navegador ou usando Curl.

```
curl https://api-id.execute-api.us-east-2.amazonaws.com
```

## Criar uma API HTTP usando a CLI da AWS

É possível usar a criação rápida para criar uma API com uma integração do Lambda ou HTTP, uma rota padrão genérica e um estágio padrão configurado para implantar alterações automaticamente. O comando [create-api](#) a seguir usa criação rápida para criar uma API que se integra a uma função do Lambda no backend.

### Note

Para invocar uma integração do Lambda, o API Gateway deve ter as permissões obrigatórias. Você pode usar uma política baseada em recursos ou uma função do IAM para conceder permissões do API Gateway para invocar uma função do Lambda. Para saber mais, consulte [AWS Lambda Permissões](#) no Guia do desenvolvedor do AWS Lambda.

### Example

```
aws apigatewayv2 create-api --name my-api --protocol-type HTTP --target
arn:aws:lambda:us-east-2:123456789012:function:function-name
```

Agora sua API está pronta para invocar. Você pode testar sua API inserindo seu URL de invocação em um navegador ou usando Curl.

```
curl https://api-id.execute-api.us-east-2.amazonaws.com
```

## Criar rotas para APIs HTTP no API Gateway

Roteia solicitações diretas recebidas de API para recursos de backend. As rotas consistem em duas partes: um método HTTP e um caminho de recurso, por exemplo, GET /pets. É possível definir métodos HTTP específicos para a rota. Ou usar o método ANY para corresponder a todos os métodos não definidos para um recurso. Você pode criar uma rota \$default que funciona como um catch-all para solicitações que não correspondem a nenhuma outra rota.

 Note

O API Gateway decodifica parâmetros de solicitação codificados em URL antes de passá-los para integrações de back-end.

## Trabalhar com variáveis de caminho

É possível usar variáveis de caminho em rotas de API HTTP.

Por exemplo, a rota GET `/pets/{petID}` captura uma solicitação GET que um cliente envia para `https://api-id.execute-api.us-east-2.amazonaws.com/pets/6`.

Uma variável de caminho voraz captura todos os recursos filho de uma rota. Para criar uma variável de caminho voraz, adicione `+` ao nome da variável, por exemplo, `{proxy+}`. O parâmetro de caminho voraz deve estar no final do caminho do recurso.

## Trabalhar com parâmetros de string de consulta

Por padrão, o API Gateway enviará parâmetros de string de consulta para a integração de backend se eles estiverem incluídos em uma solicitação para uma API Gateway.

Por exemplo, quando um cliente envia uma solicitação para `https://api-id.execute-api.us-east-2.amazonaws.com/pets?id=4&type=dog`, os parâmetros de string de consulta `?id=4&type=dog` são enviados para a integração.

## Trabalhar com a rota `$default`

A rota `$default` captura solicitações que não correspondem explicitamente a outras rotas na API.

Quando a rota `$default` recebe uma solicitação, o API Gateway envia o caminho de solicitação completo para a integração. Por exemplo, é possível criar uma API com apenas uma rota `$default` e integrá-la ao método ANY com o endpoint do HTTP `https://petstore-demo-endpoint.execute-api.com`. Quando uma solicitação é enviada para `https://api-id.execute-api.us-east-2.amazonaws.com/store/checkout`, o API Gateway envia uma solicitação para `https://petstore-demo-endpoint.execute-api.com/store/checkout`.

Para saber mais sobre integrações HTTP, consulte [Criar integrações de proxy HTTP para APIs HTTP](#).

## Rotear solicitações de API

Quando um cliente envia uma solicitação de API, o API Gateway primeiro determina para qual [estágio](#) rotear a solicitação. Se a solicitação corresponder explicitamente a um estágio, o API Gateway enviará a solicitação para esse estágio. Se nenhum estágio corresponder totalmente à solicitação, o API Gateway enviará a solicitação para o estágio `$default`. Se não houver estágio `$default`, a API retornará `{"message": "Not Found"}` e não gerará logs do CloudWatch.

Depois de selecionar um estágio, o API Gateway seleciona uma rota. Ele seleciona a rota com a correspondência mais específica, usando as seguintes prioridades:

1. Correspondência completa para uma rota e um método.
2. Correspondência para uma rota e um método com uma variável de caminho voraz (`{proxy+}`).
3. A rota `$default`.

Se nenhuma rota corresponder a uma solicitação, o API Gateway retornará `{"message": "Not Found"}` ao cliente.

Por exemplo, considere uma API com um estágio `$default` e as seguintes rotas de exemplo:

1. GET `/pets/dog/1`
2. GET `/pets/dog/{id}`
3. GET `/pets/{proxy+}`
4. ANY `/proxy+`
5. `$default`

A tabela a seguir resume como o API Gateway roteia solicitações para as rotas demonstrativas.

| Solicitação                                                                 | Rota selecionada             | Explicação                                                 |
|-----------------------------------------------------------------------------|------------------------------|------------------------------------------------------------|
| GET <code>https://api-id.execute-api.region.amazonaws.com/pets/dog/1</code> | GET <code>/pets/dog/1</code> | A solicitação corresponde totalmente a esta rota estática. |

| Solicitação                                                                      | Rota selecionada   | Explicação                                                                                                                                                   |
|----------------------------------------------------------------------------------|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GET https:// <i>api-id</i> .execute-api. <i>region</i> .amazonaws.com/pets/dog/2 | GET /pets/dog/{id} | A solicitação corresponde totalmente a essa rota.                                                                                                            |
| GET https:// <i>api-id</i> .execute-api. <i>region</i> .amazonaws.com/pets/cat/1 | GET /pets/{proxy+} | A solicitação não corresponde totalmente a uma rota. A rota com um método GET e uma variável de caminho voraz captura essa solicitação.                      |
| POST https:// <i>api-id</i> .execute-api. <i>region</i> .amazonaws.com/test/5    | ANY /{proxy+}      | O método ANY corresponde a todos os métodos não definidos para uma rota. Rotas com variáveis de caminho voraz têm prioridade mais alta que a rota \$default. |

## Tipos de endereço IP para APIs HTTP no API Gateway

Ao criar uma API, você especifica o tipo de endereços IP que podem invocar sua API. É possível escolher IPv4 para resolver endereços IPv4 e invocar sua API ou escolher pilha dupla para permitir que endereços IPv4 e IPv6 invoquem sua API. Recomendamos que você defina o tipo de endereço IP como pilha dupla para aliviar o esgotamento do espaço IP ou para seu procedimento de segurança. Para ter mais informações sobre os benefícios de um tipo de endereço IP de pilha dupla, consulte [IPv6 na AWS](#).

### Considerações sobre tipos de endereço IP

As seguintes considerações podem afetar o uso de tipos de endereço IP:

- O tipo de endereço IP padrão para APIs HTTP é IPv4.
- Se você alterar o tipo de endereço IP de uma API existente de IPv4 para pilha dupla, confirme se todas as políticas que controlam o acesso às suas APIs foram atualizadas para considerar

as chamadas IPv6. Quando você altera o tipo de endereço IP, a alteração entra em vigor imediatamente.

- Sua API pode ser associada a um nome de domínio personalizado com um tipo de endereço IP diferente do da sua API. Se você desabilitar seu endpoint de API padrão, isso poderá afetar como os chamadores podem invocar sua API.

## Alterar o tipo de endereço IP de uma API HTTP

É possível alterar o tipo de endereço IP atualizando a configuração da API. É possível atualizar a configuração da API usando o Console de gerenciamento da AWS, a AWS CLI, o CloudFormation ou um SDK da AWS. Se você alterar o tipo de endereço IP da API, não poderá reimplantá-la para que as alterações entrem em vigor.

### Console de gerenciamento da AWS

Como alterar o tipo de endereço IP de uma API HTTP

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione uma API HTTP.
3. Em Configurações da API, escolha Editar.
4. Em Tipo de endereço IP, escolha IPv4 ou Pilha dupla.
5. Escolha Salvar.

A alteração na configuração da sua API entrará em vigor imediatamente.

### AWS CLI

O comando [update-api](#) a seguir atualiza uma API para ter um tipo de endereço IP de pilha dupla:

```
aws apigatewayv2 update-api \
 --api-id abcd1234 \
 --ip-address-type dualstack
```

A saída será exibida da seguinte forma:

```
{
 "ApiEndpoint": "https://abcd1234.execute-api.us-east-1.amazonaws.com",
```

```
"ApiId": "abcd1234",
"ApiKeySelectionExpression": "$request.header.x-api-key",
"CreateDate": "2025-02-04T22:20:20+00:00",
"DisableExecuteApiEndpoint": false,
"Name": "My-HTTP-API",
"ProtocolType": "HTTP",
"RouteSelectionExpression": "$request.method $request.path",
"Tags": {},
"NotificationUri": [],
"IpAddressType": "dualstack"
}
```

## Controlar e gerenciar o acesso a APIs HTTP no API Gateway

O API Gateway é compatível com vários mecanismos de controle de acesso à sua API HTTP:

- Autorizadores do Lambda usam funções do Lambda para controlar o acesso a APIs. Para obter mais informações, consulte [Controlar o acesso a APIs HTTP com autorizadores do AWS Lambda](#).
- Os autorizadores do JWT usam tokens da Web JSON para controlar o acesso a APIs. Para obter mais informações, consulte [Controlar o acesso a APIs HTTP com autorizadores JWT no API Gateway](#).
- As funções e políticas padrão do AWS IAM oferecem controles de acesso flexíveis e robustos. É possível usar as funções e políticas do IAM para controlar quem pode criar e gerenciar suas APIs, bem como quem pode invocá-las. Para obter mais informações, consulte [Controlar o acesso a APIs HTTP com autorização do IAM no API Gateway](#).

Para melhorar seu procedimento de segurança, recomendamos que você configure um autorizador para todas as rotas na sua API HTTP. Você pode precisar fazer isso para cumprir vários requisitos de conformidade. Para ter mais informações, consulte [Amazon API Gateway controls](#) no Guia do usuário do AWS Security Hub.

### Controlar o acesso a APIs HTTP com autorizadores do AWS Lambda

Use um autorizador do Lambda para utilizar uma função do Lambda a fim de controlar o acesso à sua API HTTP. Depois, quando um cliente chamar a API, o API Gateway invocará sua função do Lambda. O API Gateway usará a resposta da função do Lambda para determinar se o cliente pode acessar sua API.



## Versão do formato da carga útil

A versão do formato de carga do autorizador especifica o formato dos dados que o API Gateway envia a um autorizador do Lambda e como o API Gateway interpreta a resposta do Lambda. Por padrão, se você não especificar uma versão de formato de carga útil, o Console de gerenciamento da AWS usará a versão mais recente. Se você criar um autorizador do Lambda usando a AWS CLI, o CloudFormation ou um SDK, será necessário especificar um `authorizerPayloadFormatVersion`. Os valores suportados são 1.0 e 2.0.

Se precisar de compatibilidade com APIs REST, use a versão 1.0.

Os exemplos a seguir mostram a estrutura de cada versão do formato de carga útil.

### 2.0

```
{
 "version": "2.0",
 "type": "REQUEST",
 "routeArn": "arn:aws:execute-api:us-east-1:123456789012:abcdef123/test/GET/request",
 "identitySource": ["user1", "123"],
 "routeKey": "$default",
 "rawPath": "/my/path",
 "rawQueryString": "parameter1=value1&parameter1=value2&parameter2=value",
 "cookies": ["cookie1", "cookie2"],
 "headers": {
 "header1": "value1",
 "header2": "value2"
 },
 "queryStringParameters": {
 "parameter1": "value1,value2",
 "parameter2": "value"
 },
 "requestContext": {
 "accountId": "123456789012",
 "apiId": "api-id",
 "authentication": {
 "clientCert": {
 "clientCertPem": "CERT_CONTENT",
 "subjectDN": "www.example.com",
 "issuerDN": "Example issuer",
 "serialNumber": "1",
 "validity": {
```

```

 "notBefore": "May 28 12:30:02 2019 GMT",
 "notAfter": "Aug 5 09:36:04 2021 GMT"
 }
},
"domainName": "id.execute-api.us-east-1.amazonaws.com",
"domainPrefix": "id",
"http": {
 "method": "POST",
 "path": "/my/path",
 "protocol": "HTTP/1.1",
 "sourceIp": "IP",
 "userAgent": "agent"
},
"requestId": "id",
"routeKey": "$default",
"stage": "$default",
"time": "12/Mar/2020:19:03:58 +0000",
"timeEpoch": 1583348638390
},
"pathParameters": { "parameter1": "value1" },
"stageVariables": { "stageVariable1": "value1", "stageVariable2": "value2" }
}

```

## 1.0

```

{
 "version": "1.0",
 "type": "REQUEST",
 "methodArn": "arn:aws:execute-api:us-east-1:123456789012:abcdef123/test/GET/request",
 "identitySource": "user1,123",
 "authorizationToken": "user1,123",
 "resource": "/request",
 "path": "/request",
 "httpMethod": "GET",
 "headers": {
 "X-AMZ-Date": "20170718T062915Z",
 "Accept": "*/*",
 "HeaderAuth1": "headerValue1",
 "CloudFront-Viewer-Country": "US",
 "CloudFront-Forwarded-Proto": "https",
 "CloudFront-Is-Tablet-Viewer": "false",

```

```
 "CloudFront-Is-Mobile-Viewer": "false",
 "User-Agent": "...",
 },
 "queryStringParameters": {
 "QueryString1": "queryValue1"
 },
 "pathParameters": {},
 "stageVariables": {
 "StageVar1": "stageValue1"
 },
 "requestContext": {
 "path": "/request",
 "accountId": "123456789012",
 "resourceId": "05c7jb",
 "stage": "test",
 "requestId": "...",
 "identity": {
 "apiKey": "...",
 "sourceIp": "...",
 "clientCert": {
 "clientCertPem": "CERT_CONTENT",
 "subjectDN": "www.example.com",
 "issuerDN": "Example issuer",
 "serialNumber": "a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1:a1",
 "validity": {
 "notBefore": "May 28 12:30:02 2019 GMT",
 "notAfter": "Aug 5 09:36:04 2021 GMT"
 }
 }
 }
 },
 "resourcePath": "/request",
 "httpMethod": "GET",
 "apiId": "abcdef123"
}
```

## Formato de resposta do autorizador do Lambda

A versão do formato de carga também determina a estrutura da resposta que deve ser retornada pela função do Lambda.

## Resposta de função do Lambda para o formato 1.0

Se você escolher a versão de formato 1.0, os autorizadores do Lambda deverão retornar uma política do IAM que permita ou negue acesso à sua rota da API. É possível usar a sintaxe de política do IAM padrão na política. Para obter políticas demonstrativas do IAM, consulte [the section called “Controlar o acesso para chamar uma API”](#). É possível transmitir propriedades de contexto para integrações do Lambda ou logs de acesso usando `$context.authorizer.property`. O objeto `context` é opcional e `claims` é um espaço reservado que não pode ser usado como objeto de contexto. Para saber mais, consulte [the section called “Variáveis 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"
 }
}
```

## Resposta de função do Lambda para o formato 2.0

Se você escolher a versão de formato 2.0, poderá retornar um valor booleano ou uma política do IAM que use a sintaxe de política padrão do IAM da sua função do Lambda. Para retornar um valor booleano, ative respostas simples para o autorizador. Os exemplos a seguir demonstram o formato que você deve codificar para ser retornado pela sua função do Lambda. O objeto `context` é opcional. É possível transmitir propriedades de contexto para integrações do Lambda ou logs de

acesso usando `$context.authorizer.property`. Para saber mais, consulte [the section called “Variáveis 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"
 }
}
```

### Funções demonstrativas do autorizador do Lambda

As funções demonstrativas do Lambda Node.js a seguir mostram os formatos de resposta que você precisa retornar de sua função do Lambda para a versão de 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

```

## Fontes de identidade

Também é possível especificar fontes de identidade para um autorizador do Lambda. As fontes de identidade especificam a localização dos dados necessários para autorizar uma solicitação. Por exemplo, é possível especificar valores de cabeçalho ou string de consulta como origens de identidade. Se você especificar fontes de identidade, os clientes deverão incluí-las na solicitação. Se a solicitação do cliente não incluir as fontes de identidade, o API Gateway não invocará o autorizador do Lambda, e o cliente receberá um erro 401.

A tabela a seguir descreve as fontes de identidade compatíveis com um autorizador do Lambda.

| Tipo                        | Exemplo                                           | Observações                                                         |
|-----------------------------|---------------------------------------------------|---------------------------------------------------------------------|
| Valor de cabeçalho          | <code>\$request.header.<i>nome</i></code>         | Nomes de cabeçalhos não diferenciam maiúsculas de minúsculas.       |
| Valor da string de consulta | <code>\$request.querystring.<i>nome</i></code>    | Os nomes de strings de consulta diferenciam maiúsculas e minúsculas |
| Variável de contexto        | <code>\$context.<i>variableName</i></code>        | O valor de uma <a href="#">variável de contexto</a> compatível.     |
| Variável de estágio         | <code>\$stageVariables.<i>variableName</i></code> | O valor de uma <a href="#">variável de estágio</a> .                |

Você também pode retornar diretamente `{"errorMessage" : "Unauthorized"}` da função do Lambda para exibir um erro 401 aos clientes. Se você retornar diretamente um erro 401 da função do Lambda para os clientes, não especifique nenhuma fonte de identidade ao criar o autorizador do Lambda.

### Armazenar em cache respostas do autorizador

É possível habilitar o armazenamento em cache para um autorizador do Lambda especificando um [authorizerResultTtlInSeconds](#). Quando o cache está habilitado para um autorizador, o API Gateway usa as fontes de identidade do autorizador como a chave de cache. Se um cliente especificar os mesmos parâmetros em fontes de identidade dentro do TTL configurado, o API Gateway usará o resultado do autorizador em cache em vez de invocar sua função do Lambda.

Para habilitar o cache, seu autorizador deve ter pelo menos uma fonte de identidade.

Se você habilitar respostas simples para um autorizador, a resposta do autorizador permitirá ou negará totalmente todas as solicitações de API que correspondam aos valores da fonte de identidade armazenados em cache. Para obter permissões mais granulares, desative respostas simples e retorne uma política do IAM. Dependendo do seu autorizador, sua política de IAM pode precisar controlar o acesso a vários recursos.

Por padrão, o API Gateway usa a resposta do autorizador em cache para todas as rotas de uma API que usam o autorizador. Para armazenar em cache as respostas por rota, adicione `$context.routeKey` às fontes de identidade do seu autorizador.

### Criar um autorizador do Lambda

Ao criar um autorizador do Lambda, especifique a função do Lambda a ser usada pelo API Gateway. É necessário conceder permissão do API Gateway para invocar a função do Lambda usando a política de recursos da função ou uma função do IAM. O comando [create-authorizer](#) indicado abaixo cria um autorizador do 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
```

O comando [add-permission](#) a seguir atualiza a política de recursos da função do Lambda para conceder permissão ao API Gateway para invocar a função. Se o API Gateway não tiver permissão para invocar sua função, os clientes receberão um 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"
```

Depois de criar um autorizador e conceder ao API Gateway permissão para invocá-lo, atualize sua rota para usar o autorizador. O comando [update-route](#) a seguir adiciona o autorizador do Lambda à rota. Se seu autorizador do Lambda usa o armazenamento em cache de política, atualize a política para controlar o acesso ao método adicional.

```
aws apigatewayv2 update-route \
 --api-id abcdef123 \
 --route-id abc123 \
 --authorization-type CUSTOM \
 --authorizer-id def123
```

## Solução de problemas em autorizadores do Lambda

Se o API Gateway não conseguir invocar o autorizador do Lambda, ou se o autorizador do Lambda retornar uma resposta em um formato inválido, os clientes receberão um arquivo `500 Internal Server Error`.

Para solucionar erros, [habilite o registro em log de acesso](#) para o estágio da API. Inclua a variável de registro em log `$context.authorizer.error` em seu formato de log.

Se os logs indicarem que o API Gateway não tem permissão para invocar sua função, atualize a política de recursos da função ou forneça uma função do IAM para conceder ao API Gateway permissão para invocar o autorizador.

Se os logs indicarem que a função do Lambda retorna uma resposta inválida, verifique se a função do Lambda retorna uma resposta no [formato necessário](#).

## Controlar o acesso a APIs HTTP com autorizadores JWT no API Gateway

O JSON Web Tokens (JWTs) pode ser usado como parte das estruturas [OpenID Connect \(OIDC\)](#) e [OAuth 2.0](#) para que você possa restringir o acesso do cliente às suas APIs.

Se você configurar um autorizador JWT para uma rota da sua API, o API Gateway valida os JWTs enviados pelos clientes com solicitações de API. O API Gateway aceita ou rejeita solicitações com base na validação de tokens e, opcionalmente, escopos no token. Se você configurar escopos para uma rota, o token deverá incluir pelo menos um dos escopos da rota.

Você pode configurar autorizadores distintos para cada rota de uma API ou usar o mesmo autorizador para várias rotas.

 Note

Não existe nenhum mecanismo padrão para diferenciar tokens de acesso JWT de outros tipos de JWTs, como os tokens OpenID e Connect ID. Exceto se precisar de tokens de ID para autorização de API, recomendamos que você configure suas rotas para exigir escopos de autorização. Você também pode configurar seus autorizadores do JWT para exigir emissores ou públicos que seu provedor de identidade usa somente ao emitir tokens de acesso do JWT.

## Autorização de solicitações de API com um autorizador JWT

O API Gateway usa o fluxo de trabalho geral a seguir para autorizar solicitações para rotas configuradas para usar um autorizador JWT.

1. Verifique se há um token em [identitySource](#). O `identitySource` pode incluir apenas o token ou o token com o prefixo `Bearer`.
2. Decodifique o token.
3. Verifique o algoritmo e a assinatura do token usando a chave pública obtida do do `emisso_jwks_uri`. No momento, somente algoritmos baseados em RSA são compatíveis. O API Gateway pode armazenar a chave pública em cache por duas horas. Como prática recomendada, ao alternar as chaves, aguarde um período de carência durante o qual as chaves antigas e novas sejam válidas.
4. Valide reivindicações. O API Gateway avalia as seguintes reivindicações de token:
  - [kid](#): o token deve ter uma reivindicação de cabeçalho que corresponda à chave no `jwks_uri` que assinou o token.
  - [iss](#): deve corresponder ao [issuer](#) configurado para o autorizador.
  - [aud](#) ou `client_id`: deve corresponder a uma das entradas [audience](#) configuradas para o autorizador. O API Gateway valida `client_id` somente se `aud` não estiver presente. Quando `aud` e `client_id` estão presentes, o API Gateway avalia `aud`.
  - [exp](#) – deve ser após a hora atual em UTC.
  - [nbf](#) – deve ser antes da hora atual em UTC.
  - [iat](#) – deve ser antes da hora atual em UTC.
  - [scope](#) ou `scp`: o token deve incluir pelo menos um dos escopos em [authorizationScopes](#) da rota.

Se qualquer uma dessas etapas falhar, o API Gateway negará a solicitação de API.

Depois de validar o JWT, o API Gateway passa as alegações no token para a integração da rota da API. Recursos de backend, como as funções do Lambda, podem acessar as alegações de JWT. Por exemplo, se o JWT incluiu um `emailID` de alegação de identidade, ele estará disponível para uma integração do Lambda em `$event.requestContext.authorizer.jwt.claims.emailID`. Para obter mais informações sobre a carga útil que o API Gateway envia para integrações do Lambda, consulte [the section called “AWS LambdaIntegrações do ”](#).

## Criar um autorizador de JWT

Antes de criar um autorizador de JWT, você deve registrar um aplicativo cliente com um provedor de identidade. Também deve ter criado uma API HTTP. Para obter exemplos de criação de uma API HTTP, consulte [Criar uma API HTTP](#).

### Criar um autorizador JWT usando o console

As etapas a seguir mostram como criar o autorizador JWT usando o console.

#### Criar um autorizador JWT usando o console

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione uma API HTTP.
3. No painel de navegação principal, selecione Autorização.
4. Escolha a guia Gerenciar autorizadores.
5. Escolha Criar.
6. Em Tipo de autorizador, escolha JWT.
7. Configure o autorizador JWT e especifique uma Origem da identidade que defina a origem do token.
8. Escolha Criar.

### Criar um autorizador JWT usando a AWS CLI

O comando [create-authorizer](#) a seguir cria um autorizador de JWT. Para `jwt-configuration`, especifique o Audience e o Issuer para seu provedor de identidades. Se você usa o Amazon Cognito como provedor de identidade, o `IssuerUrl` é `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
```

## Criar um autorizador de AWS CloudFormation

O modelo CloudFormation a seguir cria uma API HTTP com um autorizador JWT usando o Amazon Cognito como provedor de identidade.

A saída do modelo CloudFormation é um URL para uma interface de usuário hospedada no Amazon Cognito, na qual os clientes podem se inscrever e fazer login para receber um JWT. Depois que o cliente faz login, é redirecionado para sua API HTTP com um token de acesso no URL. Para invocar a API com o token de acesso, altere o # no URL para um ? para usar o token como um parâmetro de string de consulta.

## Exemplo de modelo CloudFormation

```
AWSTemplateFormatVersion: '2010-09-09'
Description: |
 Example HTTP API with a JWT authorizer. This template includes an Amazon Cognito user
 pool as the issuer for the JWT authorizer
 and an Amazon Cognito app client as the audience for the authorizer. The outputs
 include a URL for an Amazon Cognito hosted UI where clients can
 sign up and sign in to receive a JWT. After a client signs in, the client is
 redirected to your HTTP API with an access token
 in the URL. To invoke the API with the access token, change the '#' in the URL to a
 '?' to use the token as a query string parameter.

Resources:
 MyAPI:
 Type: AWS::ApiGatewayV2::Api
 Properties:
 Description: Example HTTP API
 Name: api-with-auth
 ProtocolType: HTTP
 Target: !GetAtt MyLambdaFunction.Arn
 DefaultRouteOverrides:
 Type: AWS::ApiGatewayV2::ApiGatewayManagedOverrides
 Properties:
```

```

 ApiId: !Ref MyAPI
 Route:
 AuthorizationType: JWT
 AuthorizerId: !Ref JWTAuthorizer
JWTAuthorizer:
 Type: AWS::ApiGatewayV2::Authorizer
 Properties:
 ApiId: !Ref MyAPI
 AuthorizerType: JWT
 IdentitySource:
 - '$request.querystring.access_token'
 JwtConfiguration:
 Audience:
 - !Ref AppClient
 Issuer: !Sub https://cognito-idp.${AWS::Region}.amazonaws.com/${UserPool}
 Name: test-jwt-authorizer
MyLambdaFunction:
 Type: AWS::Lambda::Function
 Properties:
 Runtime: nodejs18.x
 Role: !GetAtt FunctionExecutionRole.Arn
 Handler: index.handler
 Code:
 ZipFile: |
 exports.handler = async (event) => {
 const response = {
 statusCode: 200,
 body: JSON.stringify('Hello from the ' + event.routeKey + ' route!'),
 };
 return response;
 };
APIInvokeLambdaPermission:
 Type: AWS::Lambda::Permission
 Properties:
 FunctionName: !Ref MyLambdaFunction
 Action: lambda:InvokeFunction
 Principal: apigateway.amazonaws.com
 SourceArn: !Sub arn:${AWS::Partition}:execute-api:${AWS::Region}:
${AWS::AccountId}:${MyAPI}/${default}/${default}
FunctionExecutionRole:
 Type: AWS::IAM::Role
 Properties:
 AssumeRolePolicyDocument:
 Version: '2012-10-17'

```



```
Statement:
 - Effect: Allow
 Principal:
 Service:
 - lambda.amazonaws.com
 Action:
 - 'sts:AssumeRole'
ManagedPolicyArns:
 - arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole
UserPool:
 Type: AWS::Cognito::UserPool
 Properties:
 UserPoolName: http-api-user-pool
 AutoVerifiedAttributes:
 - email
 Schema:
 - Name: name
 AttributeDataType: String
 Mutable: true
 Required: true
 - Name: email
 AttributeDataType: String
 Mutable: false
 Required: true
AppClient:
 Type: AWS::Cognito::UserPoolClient
 Properties:
 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

```

## Atualizar uma rota para usar um autorizador JWT

Você pode usar o console, a AWS CLI ou um AWS SDK para atualizar uma rota para usar um autorizador JWT.

### Atualizar uma rota para usar um autorizador JWT com o console

As etapas a seguir mostram como atualizar uma rota para usar o autorizador JWT com o console.

Para criar um autorizador JWT usando o console

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione uma API HTTP.
3. No painel de navegação principal, selecione Autorização.
4. Escolha um método, selecione o autorizador no menu suspenso e escolha Anexar autorizador.

### Atualizar uma rota para usar um autorizador JWT com a AWS CLI

O comando [update-route](#) indicado abaixo atualiza uma rota para usar um 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

```

## Controlar o acesso a APIs HTTP com autorização do IAM no API Gateway

É possível habilitar a autorização do IAM para rotas de API HTTP. Quando a autorização do IAM está habilitada, os clientes precisam usar o [Signature Version 4 \(SigV4\)](#) para assinar suas solicitações com credenciais da AWS. O API Gateway invocará sua rota de API somente se o cliente tiver a permissão `execute-api` para a rota.

A autorização do IAM para APIs HTTP é semelhante à das [APIs REST](#).

### Note

Atualmente, as políticas de recursos não são compatíveis com as APIs HTTP.

Para obter políticas demonstrativas do IAM que concedem aos clientes permissão para invocar APIs, consulte [the section called “Controlar o acesso para chamar uma API”](#).

Habilitar a autorização do IAM para uma rota

O comando [update-route](#) indicado abaixo habilita a autorização do IAM para uma rota de API HTTP:

```
aws apigatewayv2 update-route \
 --api-id abc123 \
 --route-id abcdef \
 --authorization-type AWS_IAM
```

## Criar integrações para APIs HTTP no API Gateway

Integrações conectam uma rota aos recursos de backend. As APIs HTTP são compatíveis com integrações de proxy do Lambda, serviços da AWS e proxy HTTP. Por exemplo, é possível configurar uma solicitação POST para a rota `/signup` da sua API para se integrar a uma função do Lambda que lida com o cadastro de clientes.

### Tópicos

- [Criar integrações de proxy AWS Lambda para APIs HTTP no API Gateway](#)
- [Criar integrações de proxy HTTP para APIs HTTP](#)
- [Criar integrações de serviços da AWS para APIs HTTP no API Gateway](#)
- [Criar integrações privadas para APIs HTTP no API Gateway](#)

## Criar integrações de proxy AWS Lambda para APIs HTTP no API Gateway

Uma integração de proxy do Lambda permite integrar uma rota de API com uma função do Lambda. Quando um cliente chama sua API, o API Gateway envia a solicitação para a função do Lambda e retorna a resposta da função para o cliente. Para obter exemplos de criação de uma API HTTP, consulte [Criar uma API HTTP](#).

### Versão do formato da carga útil

A versão do formato de carga útil especifica o formato do evento que o API Gateway envia para uma integração do Lambda e como o API Gateway interpreta a resposta do Lambda. Por padrão, se você não especificar uma versão de formato de carga útil, o Console de gerenciamento da AWS usará a versão mais recente. Se você criar uma integração do Lambda usando a AWS CLI, o CloudFormation ou um SDK, será necessário especificar um `payloadFormatVersion`. Os valores suportados são 1.0 e 2.0.

Para saber mais sobre como definir `payloadFormatVersion`, consulte [create-integration](#). Para saber mais sobre como determinar a `payloadFormatVersion` de uma integração existente, consulte [get-integration](#).

### Diferenças do formato da carga útil

A lista a seguir mostra as diferenças entre as versões do formato da carga útil 1.0 e 2.0:

- O formato 2.0 não tem campos `multiValueHeaders` ou `multiValueQueryStringParameters`. Os cabeçalhos duplicados são combinados com vírgulas e incluídos no campo `headers`. As strings de consulta duplicadas são combinadas com vírgulas e incluídas no campo `queryStringParameters`.
- O formato 2.0 tem `rawPath`. Se você usar um mapeamento de API para associar o estágio a um nome de domínio personalizado, `rawPath` não fornecerá o valor do mapeamento de API. Use o formato 1.0 e `path` para acessar o mapeamento da API de seu nome de domínio personalizado.
- O formato 2.0 inclui um novo campo `cookies`. Todos os cabeçalhos de cookie na solicitação são combinados com vírgulas e adicionados ao campo `cookies`. Na resposta ao cliente, cada cookie torna-se um cabeçalho `set-cookie`.

### Estrutura do formato da carga útil

Os exemplos a seguir mostram a estrutura de cada versão do formato de carga útil. Todos os nomes de cabeçalho ficam em letras 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",
 "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 resposta da função do Lambda

A versão do formato da carga determina a estrutura da resposta que a função do Lambda deve retornar.

### Resposta de função do Lambda para o formato 1.0

Com a versão do formato 1.0, as integrações do Lambda devem retornar uma resposta no formato JSON a seguir:

#### Example

```

{
 "isBase64Encoded": true|false,
 "statusCode": httpStatusCode,
 "headers": { "headername": "headervalue", ... },
 "multiValueHeaders": { "headername": ["headervalue", "headervalue2", ...], ... },
 "body": "..."
}

```



```
}
```

## Resposta de função do Lambda para o formato 2.0

Com a versão de formato 2.0, o API Gateway pode inferir o formato de resposta para você. O API Gateway fará as suposições a seguir se sua função do Lambda retornar JSON válido e não retornar um `statusCode`:

- `isBase64Encoded` é `false`.
- `statusCode` é `200`.
- `content-type` é `application/json`.
- `body` é a resposta da função.

Os exemplos a seguir mostram a saída de uma função do Lambda e a interpretação do API Gateway.

| Saída da função do Lambda                      | Interpretação do API Gateway                                                                                                                                                    |
|------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>"Hello from Lambda!"</pre>                | <pre>{   "isBase64Encoded": false,   "statusCode": 200,   "body": "Hello from Lambda!",   "headers": {     "content-type": "application/ json"   } }</pre>                      |
| <pre>{ "message": "Hello from Lambda!" }</pre> | <pre>{   "isBase64Encoded": false,   "statusCode": 200,   "body": "{ \"message\": \"Hello from Lambda!\" }",   "headers": {     "content-type": "application/ json"   } }</pre> |

Para personalizar a resposta, a função do Lambda deve retornar uma resposta com o formato a seguir.

```
{
 "cookies" : ["cookie1", "cookie2"],
 "isBase64Encoded": true|false,
 "statusCode": httpStatusCode,
 "headers": { "headername": "headervalue", ... },
 "body": "Hello from Lambda!"
}
```

## Criar integrações de proxy HTTP para APIs HTTP

Uma integração de proxy HTTP permite que você conecte uma rota de API a um endpoint HTTP roteável publicamente. Com esse tipo de integração, o API Gateway transmite toda a solicitação e resposta entre o front-end e o backend.

Para criar uma integração de proxy HTTP, forneça a URL de um endpoint HTTP roteável publicamente.

### Integração de proxy HTTP com variáveis de caminho

É possível usar variáveis de caminho em rotas de API HTTP.

Por exemplo, a rota `/pets/{petID}` captura solicitações para a `/pets/6`. Você pode fazer referência a variáveis de caminho no URI de integração para enviar o conteúdo das variáveis para uma integração. Um exemplo é `/pets/extendedpath/{petID}`.

Você pode usar variáveis de caminho voraz para capturar todos os recursos filho de uma rota. Para criar uma variável de caminho voraz, adicione `+` ao nome da variável, por exemplo, `{proxy+}`.

Para configurar uma rota com uma integração de proxy HTTP que captura todas as solicitações, crie uma rota de API com uma variável de caminho voraz (por exemplo, `/parent/{proxy+}`). Integre a rota a um endpoint HTTP (por exemplo, `https://petstore-demo-endpoint.execute-api.com/petstore/{proxy}`) no método ANY. O parâmetro de caminho voraz deve estar no final do caminho do recurso.

## Criar integrações de serviços da AWS para APIs HTTP no API Gateway

Você pode integrar sua API HTTP com serviços da AWS usando integrações de primeira classe. Uma integração de primeira classe conecta uma rota de API HTTP a uma API de serviço da AWS. Quando um cliente invoca uma rota apoiada por uma integração de primeira classe, o API Gateway

invoca uma API de serviço da AWS para você. Por exemplo, é possível usar integrações de primeira classe para enviar uma mensagem para uma fila do Amazon Simple Queue Service ou iniciar uma máquina de estado do AWS Step Functions. Para obter ações de serviço compatíveis, consulte [the section called “AWSReferência de integrações de serviço da ”](#).

### Mapear parâmetros de solicitação

Integrações de primeira classe têm parâmetros obrigatórios e opcionais. É necessário configurar todos os parâmetros necessários para criar uma integração. Você pode usar valores estáticos ou mapear parâmetros que são avaliados dinamicamente no tempo de execução. Para obter uma lista completa de integrações e parâmetros compatíveis, consulte [the section called “AWSReferência de integrações de serviço da ”](#).

A tabela a seguir descreve os parâmetros de solicitação de mapeamento compatíveis.

| Tipo                        | Exemplo                                        | Observações                                                                                                                                                             |
|-----------------------------|------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valor de cabeçalho          | <code>\$request.header.<i>nome</i></code>      | Nomes de cabeçalhos não diferenciam maiúsculas de minúsculas. O API Gateway combina vários valores de cabeçalho com vírgulas, por exemplo, "header1": "value1,value2" . |
| Valor da string de consulta | <code>\$request.querystring.<i>nome</i></code> | Os nomes de strings de consulta diferenciam maiúsculas e minúsculas. O API Gateway combina vários valores com vírgulas, por exemplo, "querystring1": "Value1,Value2" .  |
| Parâmetro de caminho        | <code>\$request.path.<i>nome</i></code>        | O valor de um parâmetro de caminho na solicitação. Por exemplo, se a rota for <code>/pets/{petId}</code> , você poderá mapear o parâmetro                               |

| Tipo                        | Exemplo                                    | Observações                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-----------------------------|--------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                             |                                            | petId da solicitação com <code>\$request.path.petId</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Solicitar passagem do corpo | <code>\$request.body</code>                | O API Gateway transmite todo o corpo da solicitação.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Corpo da solicitação        | <code>\$request.body.name</code>           | <p>Uma <a href="#">expressão de caminho JSON</a>. Descidas recursivas (<code>\$request.body..name</code>) e expressões de filtro (<code>(expression)</code>) não são compatíveis.</p> <div> <p> <b>Note</b></p> <p>Quando você especifica um caminho JSON, o API Gateway trunca o corpo da solicitação em 100 KB e, em seguida, aplica a expressão de seleção. Para enviar cargas maiores que 100 KB, especifique <code>\$request.body</code> .</p> </div> |
| Variável de contexto        | <code>\$context.variableName</code>        | O valor de uma <a href="#">variável de contexto</a> compatível.                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Variável de estágio         | <code>\$stageVariables.variableName</code> | O valor de uma <a href="#">variável de estágio</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Valor estático              | <code>string</code>                        | Um valor constante.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

## Criar uma integração de primeira classe

Antes de criar uma integração de primeira classe, é necessário criar uma função do IAM que conceda ao API Gateway permissões para invocar a ação de serviço da AWS à qual você está se integrando. Para saber mais, consulte [Criação de uma função para um serviço da AWS](#).

Para criar uma integração de primeira classe, escolha uma ação de serviço da AWS compatível, como SQS-SendMessage, configure os parâmetros de solicitação e forneça uma função que conceda ao API Gateway permissões para invocar a API de serviço integrado da AWS. Dependendo do subtipo de integração, diferentes parâmetros de solicitação são necessários. Para saber mais, consulte [the section called “AWSReferência de integrações de serviço da”](#).

O comando [create-integration](#) indicado abaixo cria uma integração que envia uma mensagem do 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"}'
```

## Criar uma integração de primeira classe usando o CloudFormation

O exemplo a seguir mostra um trecho do CloudFormation que cria uma rota `/ {source} / {detailType}` com uma integração de primeira classe com o Amazon EventBridge.

O parâmetro `Source` é mapeado para o parâmetro de caminho `{source}`, o `DetailType` é mapeado para o parâmetro de caminho `{DetailType}` e o parâmetro `Detail` é mapeado para o corpo da solicitação.

O trecho não mostra o barramento de eventos ou o perfil do IAM que concede permissões ao API Gateway para invocar a ação `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"
```

## Referência do subtipo de integração

Os [subtipos de integração](#) a seguir são compatíveis com APIs HTTP.

### Subtipos de integração

- [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

Envia eventos personalizados para o Amazon EventBridge para que eles possam ser correspondidos a regras.

| Parâmetro    | Obrigatório |
|--------------|-------------|
| Detalhes     | Verdadeiro  |
| DetailType   | Verdadeiro  |
| Origem       | Verdadeiro  |
| Tempo        | Falso       |
| EventBusName | Falso       |
| Recursos     | Falso       |
| Região       | Falso       |
| TraceHeader  | Falso       |

Para saber mais, consulte [PutEvents](#) na Referência de API do Amazon EventBridge.

### SQS-SendMessage 1.0

Entrega uma mensagem para a fila especificada.

| Parâmetro               | Obrigatório |
|-------------------------|-------------|
| QueueUrl                | Verdadeiro  |
| MessageBody             | Verdadeiro  |
| DelaySeconds            | Falso       |
| MessageAttributes       | Falso       |
| MessageDeduplicationId  | Falso       |
| MessageGroupId          | Falso       |
| MessageSystemAttributes | Falso       |
| Região                  | Falso       |

Para saber mais, consulte [SendMessage](#) na Referência de API do Amazon Simple Queue Service.

### SQS-ReceiveMessage 1.0

Recupera uma ou mais mensagens (até 10) de uma fila especificada.

| Parâmetro               | Obrigatório |
|-------------------------|-------------|
| QueueUrl                | Verdadeiro  |
| AttributeNames          | Falso       |
| MaxNumberOfMessages     | Falso       |
| MessageAttributeNames   | Falso       |
| ReceiveRequestAttemptId | Falso       |
| VisibilityTimeout       | Falso       |
| WaitTimeSeconds         | Falso       |
| Região                  | Falso       |

Para saber mais, consulte [ReceiveMessage](#) na Referência de API do Amazon Simple Queue Service.

### SQS-DeleteMessage 1.0

Exclui a mensagem especificada da fila especificada.

| Parâmetro     | Obrigatório |
|---------------|-------------|
| ReceiptHandle | Verdadeiro  |
| QueueUrl      | Verdadeiro  |
| Região        | Falso       |

Para saber mais, consulte [DeleteMessage](#) na Referência de API do Amazon Simple Queue Service.



## SQS-PurgeQueue 1.0

Exclui todas as mensagens da fila especificada.

| Parâmetro | Obrigatório |
|-----------|-------------|
| QueueUrl  | Verdadeiro  |
| Região    | Falso       |

Para saber mais, consulte [PurgeQueue](#) na Referência de API do Amazon Simple Queue Service.

## AppConfig-GetConfiguration 1.0

Receba informações sobre uma configuração.

| Parâmetro                  | Obrigatório |
|----------------------------|-------------|
| Aplicativo                 | Verdadeiro  |
| Ambiente                   | Verdadeiro  |
| Configuração               | Verdadeiro  |
| ClientId                   | Verdadeiro  |
| ClientConfigurationVersion | Falso       |
| Região                     | Falso       |

Para saber mais, consulte [GetConfiguration](#) na Referência da API AppConfig da AWS.

## Kinesis-PutRecord 1.0

Grava um único registro de dados em um stream de dados do Amazon Kinesis.

| Parâmetro  | Obrigatório |
|------------|-------------|
| StreamName | Verdadeiro  |

| Parâmetro                 | Obrigatório |
|---------------------------|-------------|
| Dados                     | Verdadeiro  |
| PartitionKey              | Verdadeiro  |
| SequenceNumberForOrdering | Falso       |
| ExplicitHashKey           | Falso       |
| Região                    | Falso       |

Para saber mais, consulte [PutRecord](#) na Referência de API do Amazon Kinesis Data Streams.

#### StepFunctions-StartExecution 1.0

Inicia uma execução de máquina de estado.

| Parâmetro       | Obrigatório |
|-----------------|-------------|
| StateMachineArn | Verdadeiro  |
| Nome            | Falso       |
| Entrada         | Falso       |
| Região          | Falso       |

Para saber mais, consulte [StartExecution](#) na Referência da API do AWS Step Functions.

#### StepFunctions-StartSyncExecution 1.0

Inicia uma execução de máquina de estado síncrono.

| Parâmetro       | Obrigatório |
|-----------------|-------------|
| StateMachineArn | Verdadeiro  |
| Nome            | Falso       |

| Parâmetro   | Obrigatório |
|-------------|-------------|
| Entrada     | Falso       |
| Região      | Falso       |
| TraceHeader | Falso       |

Para saber mais, consulte [StartSyncExecution](#) na Referência da API do AWS Step Functions.

### StepFunctions-StopExecution 1.0

Interrompe uma execução.

| Parâmetro    | Obrigatório |
|--------------|-------------|
| ExecutionArn | Verdadeiro  |
| Causa        | Falso       |
| Erro         | Falso       |
| Região       | Falso       |

Para saber mais, consulte [StopExecution](#) na Referência da API do AWS Step Functions.

## Criar integrações privadas para APIs HTTP no API Gateway

As integrações privadas permitem que você crie integrações de API com recursos privados em uma VPC, como Application Load Balancers ou aplicações baseadas em contêiner do Amazon ECS.

Você pode expor seus recursos em uma VPC para acesso por clientes fora da VPC usando integrações privadas. Você pode controlar o acesso à sua API usando qualquer um dos [métodos de autorização](#) compatíveis com o API Gateway.

**Note**

Para criar uma integração privada, primeiro crie um link de VPC. Os links da VPC V2 agora comportam APIs HTTP e REST. Para saber mais sobre links da VPC V2, consulte [the section called “Links de VPC V2”](#).

Depois de criar um link da VPC V2, é possível configurar integrações privadas que se conectam a um Application Load Balancer, Network Load Balancer ou a recursos registrados em um serviço do AWS Cloud Map.

**Considerações**

As considerações a seguir podem afetar o uso de integrações privadas:

- Todos os recursos devem ser de propriedade da mesma Conta da AWS. Isso inclui o balanceador de carga ou o serviço AWS Cloud Map, o link da VPC e a API HTTP.
- Por padrão, o tráfego de integração privada usa o protocolo HTTP. Para usar HTTPS, especifique uma [tlsConfig](#). Para fazer isso usando o Console de gerenciamento da AWS, ao criar sua integração privada, escolha Configurações avançadas e, depois, insira um nome de servidor seguro.
- Para integrações privadas, o API Gateway inclui a parte do [estágio](#) do endpoint da API na solicitação para seus recursos de backend. Por exemplo, uma solicitação para o estágio test de uma API inclui test/*route-path* na solicitação para sua integração privada. Para remover o nome do estágio da solicitação para os seus recursos de backend, use o [mapeamento de parâmetros](#) para substituir o caminho da solicitação para `$request.path`.

Criar uma integração privada usando um Application Load Balancer ou Network Load Balancer

Antes de criar uma integração privada, você deve criar um link da VPC V2. Para saber mais sobre links da VPC V2, consulte [the section called “Links de VPC V2”](#).

Para criar uma integração privada com um Application Load Balancer ou Network Load Balancer, crie uma integração de proxy HTTP, especifique o link de VPC a ser usado e forneça o ARN de listener do balanceador de carga.

O comando [create-integration](#) indicado abaixo cria uma integração privada que se conecta a um balanceador de carga usando um link de 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
```

## Criar uma integração privada usando a descoberta de serviço do AWS Cloud Map

Antes de criar uma integração privada, você deve criar um link da VPC V2. Para saber mais sobre links de VPC, consulte [the section called “Links de VPC V2”](#).

Para integrações com o AWS Cloud Map, o API Gateway usa `DiscoverInstances` para identificar recursos. Você pode usar parâmetros de consulta para direcionar recursos específicos. Os atributos dos recursos registrados devem incluir endereços IP e portas. O API Gateway distribui solicitações entre recursos íntegros que são retornados de `DiscoverInstances`. Para saber mais, consulte [DiscoverInstances](#) na Referência da API do AWS Cloud Map.

### Note

Se usar o Amazon ECS para preencher entradas no AWS Cloud Map, você deverá configurar a tarefa do Amazon ECS para usar registros SRV com a descoberta de serviços do Amazon ECS ou ativar o Amazon ECS Service Connect. Para obter mais informações, consulte [Interconexão de serviços](#) no Guia do desenvolvedor do Amazon Elastic Container Service.

Para criar uma integração privada com o AWS Cloud Map, crie uma integração de proxy HTTP, especifique o link de VPC a ser usado e forneça o ARN do serviço AWS Cloud Map.

O comando [create-integration](#) indicado abaixo cria uma integração privada que usa a descoberta de serviços do 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
```

## Configurar CORS para APIs HTTP no API Gateway

O [compartilhamento de recursos entre origens \(CORS\)](#) é um recurso de segurança de navegador que restringe as solicitações HTTP que são iniciadas em scripts em execução no navegador. Se você não conseguir acessar sua API e receber uma mensagem de erro contendo `Cross-Origin Request Blocked`, talvez seja necessário habilitar o CORS. Consulte mais informações em [O que é CORS?](#).

Normalmente o CORS é necessário para criar aplicativos Web que acessem APIs hospedadas em um domínio ou origem diferente. O CORS pode ser habilitado para permitir solicitações para sua API a partir de um aplicativo Web hospedado em um domínio diferente. Por exemplo, se sua API estiver hospedada em `https://{api_id}.execute-api.{region}.amazonaws.com/`, e você quiser chamar sua API a partir de um aplicativo web hospedado no `example.com`, sua API deverá ser compatível com CORS.

Se você configurar o CORS para uma API, o API Gateway enviará automaticamente uma resposta às solicitações de comprovação `OPTIONS`, mesmo que não haja uma rota `OPTIONS` configurada para sua API. Para uma solicitação CORS, o API Gateway adiciona os cabeçalhos de CORS configurados à resposta de uma integração.

### Note

Se você configurar CORS para uma API, o API Gateway ignorará os cabeçalhos de CORS retornados de sua integração de backend.

Em uma configuração de CORS, é possível especificar os parâmetros a seguir. Para adicionar esses parâmetros usando o console da API HTTP do API Gateway, selecione Adicionar depois de inserir o valor.

| Cabeçalhos de CORS          | Propriedade da configuração de CORS | Exemplos de valores                                                                                                                     |
|-----------------------------|-------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| Access-Control-Allow-Origin | allowOrigins                        | <ul style="list-style-type: none"><li><code>https://www.example.com</code></li><li><code>*</code> (permitir todas as origens)</li></ul> |

| Cabeçalhos de CORS               | Propriedade da configuração de CORS | Exemplos de valores                                                                                                                                                                |
|----------------------------------|-------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                  |                                     | <ul style="list-style-type: none"> <li>https://* (permitir qualquer origem que comece com https://)</li> <li>http://* (permitir qualquer origem que comece com http://)</li> </ul> |
| 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 retornar cabeçalhos CORS, sua solicitação deve conter um cabeçalho `origin`. Para o método `OPTIONS`, sua solicitação deve conter um cabeçalho `origin` e um cabeçalho `Access-Control-Request-Method`.

A configuração de CORS pode parecer com a seguinte imagem:

API Gateway

APIs

Custom domain names

VPC links

API: HelloWorld...(abcd1234)

▼ Develop

Routes

Authorization

Integrations

CORS

Reimport

Export

▼ Deploy

Stages

▼ Monitor

Metrics

Logging

▼ Protect

Throttling

API Gateway > APIs > HelloWorld (abcd1234) > CORS

### Cross-Origin Resource Sharing

#### Configure CORS [Info](#)

CORS allows resources from different domains to be loaded by browsers. If you configure CORS for an API, API Gateway ignores CORS headers returned from your backend integration. See our [CORS documentation](#) for more details.

Access-Control-Allow-Origin

Enter a value for Allowed Origins

Add

https://www.example.com X

Access-Control-Allow-Headers

Enter a value for Allowed Headers

Add

authorization X

Access-Control-Allow-Methods

Choose Allowed Methods

\*

Access-Control-Expose-Headers

Enter a value for Exposed Headers

Add

date,x-api-id X

Access-Control-Max-Age

300

Access-Control-Allow-Credentials

☒ YES

Cancel

Save

Configurar o CORS para uma API HTTP com uma rota **\$default** e um autorizador

É possível habilitar o CORS e configurar a autorização para qualquer rota de uma API HTTP. Quando você habilita o CORS e a autorização para a [rota \\$default](#), há algumas considerações especiais. A rota `$default` captura solicitações para todos os métodos e rotas não definidos explicitamente, inclusive solicitações `OPTIONS`. Para oferecer suporte a solicitações `OPTIONS` não autorizadas, adicione uma rota `OPTIONS /{proxy+}` à API que não exija autorização e anexe uma integração à rota. A rota `OPTIONS /{proxy+}` tem prioridade mais alta que a `$default`. Como resultado, ela permite que os clientes enviem solicitações `OPTIONS` para a API sem autorização. Para obter mais informações sobre prioridades de roteamento, consulte [Rotear solicitações de API](#).

Configurar o CORS para uma API HTTP usando a CLI da AWS

O comando [update-api](#) a seguir habilita as solicitações de CORS em `https://www.example.com`.

CORS

1257



## Example

```
aws apigatewayv2 update-api --api-id api-id --cors-configuration AllowOrigins="https://www.example.com"
```

Para obter mais informações, consulte [CORS](#) na Referência de API do Amazon API Gateway versão 2.

## Transformar solicitações e respostas de API para APIs HTTP no API Gateway

Você pode modificar solicitações de API dos clientes antes que eles atinjam suas integrações de backend. Você também pode alterar a resposta das integrações antes que o API Gateway retorne a resposta aos clientes. Use o mapeamento de parâmetros para modificar solicitações e respostas de API para APIs HTTP. Para usar o mapeamento de parâmetros, especifique os parâmetros de solicitação de API ou resposta a serem modificados e especifique como modificar esses parâmetros.

### Transformação de solicitações de API

Você usa parâmetros de solicitação para alterar solicitações antes que elas atinjam suas integrações de backend. Você pode modificar cabeçalhos, strings de consulta ou o caminho da solicitação.

Os parâmetros de solicitação são um mapa de chave-valor. A chave identifica a localização do parâmetro da solicitação a ser alterado e como alterá-lo. O valor especifica os novos dados para o parâmetro.

A tabela a seguir mostra as chaves suportadas.

| Tipo               | Sintaxe                                                      |
|--------------------|--------------------------------------------------------------|
| Cabeçalho          | append overwrite remove:header. <i>headername</i>            |
| String de consulta | append overwrite remove:querystring. <i>querystring-name</i> |
| Caminho            | overwrite:path                                               |

A tabela a seguir mostra os valores suportados que você pode mapear para parâmetros.

| Tipo                        | Sintaxe                                                                                            | Observações                                                                                                                                                                                                                                                                                      |
|-----------------------------|----------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valor de cabeçalho          | <code>\$request.header.<i>name</i></code> ou <code>\${request.header.<i>name</i>}</code>           | Nomes de cabeçalhos não diferenciam maiúsculas de minúsculas. O API Gateway combina vários valores de cabeçalho com vírgulas, por exemplo, "header1": "value1,value2" . Alguns cabeçalhos são reservados. Para saber mais, consulte <a href="#">the section called “Cabeçalhos reservados”</a> . |
| Valor da string de consulta | <code>\$request.querystring.<i>name</i></code> ou <code>\${request.querystring.<i>name</i>}</code> | Os nomes de strings de consulta diferenciam maiúsculas e minúsculas. O API Gateway combina vários valores com vírgulas, por exemplo, "querystring1" "Value1,Value2" .                                                                                                                            |
| Corpo da solicitação        | <code>\$request.body.<i>name</i></code> ou <code>\${request.body.<i>name</i>}</code>               | Uma expressão de caminho JSON. Descida recursiva ( <code>\$request.body..name</code> e) e expressões de filtro ( <code>(expression)</code> ) não são suportadas.                                                                                                                                 |

 **Note**  
Quando você especifica um caminho JSON, o API Gateway trunca o corpo da solicitação em 100 KB e,

| Tipo                      | Sintaxe                                                                                    | Observações                                                                                                                                                                                                                                                |
|---------------------------|--------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                           |                                                                                            | em seguida, aplica a expressão de seleção. Para enviar cargas maiores que 100 KB, especifique <code>\$request.body</code> .                                                                                                                                |
| O caminho da solicitação. | <code>\$request.path</code> ou <code>\${request.path}</code>                               | O caminho da solicitação, sem o nome do estágio.                                                                                                                                                                                                           |
| Parâmetro de caminho      | <code>\$request.path.name</code> ou <code>\${request.path.name}</code>                     | O valor de um parâmetro de caminho na solicitação. Por exemplo, se a rota for <code>/pets/{petId}</code> , você poderá mapear o parâmetro <code>petId</code> da solicitação com <code>\$request.path.petId</code> .                                        |
| Variável de contexto      | <code>\$context.variableName</code> ou <code>\${context.variableName}</code>               | O valor de uma <a href="#">variável de contexto</a> .<br><br><div> <b>Note</b><br/>Somente os caracteres especiais <code>.</code> e <code>_</code> são aceitos.</div> |
| Variável de estágio       | <code>\$stageVariables.variableName</code> ou <code>\${stageVariables.variableName}</code> | O valor de uma <a href="#">variável de estágio</a> .                                                                                                                                                                                                       |
| Valor estático            | <code>string</code>                                                                        | Um valor constante.                                                                                                                                                                                                                                        |

**Note**

Para usar mais de uma variável em uma expressão de seleção, inclua a variável entre colchetes. Por exemplo, `${request.path.name} ${request.path.id}`.

## Transformando respostas da API

Você usa parâmetros de resposta para transformar a resposta HTTP de uma integração de backend antes de retornar a resposta aos clientes. Você pode modificar cabeçalhos ou o código de status de uma resposta antes que o API Gateway retorne a resposta aos clientes.

Você configura os parâmetros de resposta para cada código de status que sua integração retorna. Os parâmetros de resposta são um mapa de chave-valor. A chave identifica a localização do parâmetro da solicitação a ser alterado e como alterá-lo. O valor especifica os novos dados para o parâmetro.

A tabela a seguir mostra as chaves suportadas.

| Tipo             | Sintaxe                                                        |
|------------------|----------------------------------------------------------------|
| Cabeçalho        | <code>append overwrite remove:header. <i>headername</i></code> |
| Código de status | <code>overwrite:statuscode</code>                              |

A tabela a seguir mostra os valores suportados que você pode mapear para parâmetros.

| Tipo               | Sintaxe                                                                                    | Observações                                                                                                                                                                                 |
|--------------------|--------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Valor de cabeçalho | <code>\$response.header.<i>nome</i></code> ou <code>\${response.header.<i>nome</i>}</code> | Nomes de cabeçalhos não diferenciam maiúsculas de minúsculas. O API Gateway combina vários valores de cabeçalho com vírgulas, por exemplo, <code>"header1": "value1,value2"</code> . Alguns |

| Tipo                 | Sintaxe                                                                                    | Observações                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|----------------------|--------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                      |                                                                                            | cabeçalhos são reservados. Para saber mais, consulte <a href="#">the section called “Cabeçalhos reservados”</a> .                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Corpo da resposta    | <code>\$response.body.name</code> ou <code>\${response.body.name}</code>                   | <p>Uma expressão de caminho JSON. Descidas recursivas (<code>\$response.body.name</code>) e expressões de filtro (<code>(expression)</code>) não são compatíveis.</p> <div>  <b>Note</b><br/>           Quando você especifica um caminho JSON, o API Gateway trunca o corpo da resposta a 100 KB e, em seguida, aplica a expressão de seleção. Para enviar cargas maiores que 100 KB, especifique <code>\$response.body</code>.         </div> |
| Variável de contexto | <code>\$context.variableName</code> ou <code>\${context.variableName}</code>               | O valor de uma <a href="#">variável de contexto</a> compatível.                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Variável de estágio  | <code>\$stageVariables.variableName</code> ou <code>\${stageVariables.variableName}</code> | O valor de uma <a href="#">variável de estágio</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| Valor estático       | <i>string</i>                                                                              | Um valor constante.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

 Note

Para usar mais de uma variável em uma expressão de seleção, inclua a variável entre colchetes. Por exemplo, `${request.path.name} ${request.path.id}`.

## Cabeçalhos reservados

Os seguintes cabeçalhos são reservados. Não é possível configurar mapeamentos de solicitação ou resposta para esses cabeçalhos.

- access-control-\*
- apigw-\*
- Autorização
- Conexão
- Content-Encoding
- Content-Length
- Content-Location
- Encaminhado
- Keep-alive
- Origem
- Proxy-Authenticate
- Proxy-Authorization
- TE
- Trailers
- Transfer-Encoding
- Upgrade
- x-amz-\*
- x-amzn-\*
- X-Forwarded-For
- X-Forwarded-Host
- X-Forwarded-Proto
- Via

## Exemplos

Os exemplos da AWS CLI a seguir configuram mapeamentos de dados. Para ver modelos de exemplo do CloudFormation, consulte o [GitHub](#).

### Adicionar um cabeçalho a uma solicitação de API

O comando [create-integration](#) a seguir cria um cabeçalho chamado `header1` para uma solicitação de API antes que ela atinja sua integração de backend. O API Gateway preenche o cabeçalho com o ID da solicitação.

```
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" }'
```

### Renomear um cabeçalho de solicitação

O comando [create-integration](#) indicado abaixo renomeia um cabeçalho de solicitação de `header1` para `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": "" }'
```

### Alterar a resposta de uma integração

O comando [create-integration](#) a seguir configura parâmetros de resposta para uma integração. Quando as integrações retornam um código de status 500, o API Gateway altera o código de status para 403 e adiciona `header11` à resposta. Quando a integração retorna um código de status 404, o API Gateway adiciona um erro cabeçalho à resposta.

```
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": "" }'
```

```
--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"} }'
```

## Remover mapeamentos de parâmetros configurados

O comando [update-integration](#) a seguir remove os parâmetros de solicitação configurados anteriormente para `append:header.header1`. Ele também remove parâmetros de resposta previamente configurados para um código de status 200.

```
aws apigatewayv2 update-integration \
--api-id abcdef123 \
--integration-id hijk456 \
--request-parameters '{"append:header.header1" : ""}' \
--response-parameters '{"200" : {}}'
```

## Usar definições do OpenAPI para APIs HTTP no API Gateway

É possível definir a API HTTP usando um arquivo de definição de OpenAPI 3.0. Depois, é possível importar a definição no API Gateway para criar uma API. Para saber mais sobre as extensões do API Gateway para OpenAPI, consulte [Extensões do OpenAPI](#).

### Importar uma API HTTP

É possível criar uma API HTTP importando um arquivo de definição do OpenAPI 3.0.

Para realizar a migração de uma API REST para uma API HTTP, é possível exportar sua API REST como um arquivo de definição do OpenAPI 3.0. Depois, importe a definição de API como uma API HTTP. Para saber mais sobre como exportar uma API REST, consulte [Exportar uma API REST do API Gateway](#).

#### Note

As APIs HTTP são compatíveis com as mesmas variáveis da AWS que as APIs REST. Para saber mais, consulte [AWS de variáveis para importação de OpenAPI](#).



## Importar informações de validação

À medida que você importa uma API, o API Gateway fornece três categorias de informações de validação.

### Informações

Uma propriedade é válida de acordo com a especificação do OpenAPI, mas essa propriedade não é compatível com APIs HTTP.

Por exemplo, o snippet do OpenAPI 3.0 a seguir produz informações sobre importação porque APIs HTTP não são compatíveis com a validação de solicitação. O API Gateway ignora os campos `requestBody` e `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"
 }
 }
 }
 }
}
```

```

 }
 }
}
...
}
...
}

```

## Aviso

Uma propriedade ou estrutura é inválida de acordo com a especificação OpenAPI, mas não bloqueia a criação de API. É possível especificar se o API Gateway deve ignorar esses avisos e continuar criando a API ou parar de criar a API em avisos.

O documento do OpenAPI 3.0 a seguir produz avisos na importação porque APIs HTTP são compatíveis apenas com integrações de proxy do Lambda e HTTP.

```

"x-amazon-apigateway-integration": {
 "type": "AWS",
 "httpMethod": "POST",
 "uri": "arn:aws:lambda:us-east-2:123456789012:function:HelloWorld",
 "payloadFormatVersion": "1.0"
}

```

## Erro

A especificação do OpenAPI é inválida ou malformada. O API Gateway não pode criar recursos a partir do documento malformado. Você precisa corrigir os erros e, em seguida, tentar novamente.

A definição de API a seguir produz erros na importação porque as APIs HTTP são compatíveis apenas com a especificação do OpenAPI 3.0.

```

{
 "swagger": "2.0.0",
 "info": {
 "title": "My API",
 "description": "An Example OpenAPI definition for Errors/Warnings/ImportInfo",
 "version": "1.0"
 }
 ...
}

```

Como outro exemplo, enquanto o OpenAPI permite que os usuários definam uma API com vários requisitos de segurança anexados a uma determinada operação, o API Gateway não oferece suporte a isso. Cada operação pode ter apenas uma autorização do IAM, um autorizador do Lambda ou um autorizador JWT. A tentativa de modelar vários requisitos de segurança resulta em um erro.

## Importar uma API usando a AWS CLI

O comando [import-api](#) indicado abaixo importa o arquivo de definição `api-definition.json` do OpenAPI 3.0 como uma API HTTP:

### Example

```
aws apigatewayv2 import-api --body file://api-definition.json
```

### Example

É possível importar a definição demonstrativa do OpenAPI 3.0 a seguir para criar uma 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"
]
}
}
```

## Exportar APIs HTTP do API Gateway

Depois de criar uma API HTTP, é possível exportar uma definição do OpenAPI 3.0 da API a partir do API Gateway. Você pode escolher um estágio para exportar ou exportar a configuração mais recente da API. Também é possível importar uma definição de API exportada para o API Gateway para a criação de outra API idêntica. Para saber mais sobre a importação de definições de API, consulte [Importar uma API HTTP](#).

Exportar uma definição do OpenAPI 3.0 de um estágio usando a CLI da AWS

O comando [export-api](#) a seguir exporta uma definição do OpenAPI de um estágio de API chamado prod para um arquivo YAML denominado stage-definition.yaml. O arquivo de definição exportado inclui [extensões do API Gateway](#) por padrão.

```
aws apigatewayv2 export-api \
 --api-id api-id \
 --output-type YAML \
 --specification OAS30 \
 --stage-name prod \
 stage-definition.yaml
```

Exporte uma definição do OpenAPI 3.0 das alterações mais recentes da API usando a CLI da AWS

O comando [export-api](#) a seguir exporta uma definição do OpenAPI de uma API HTTP para um arquivo JSON denominado latest-api-definition.json. Como o comando não especifica um estágio, o API Gateway exporta a configuração mais recente da API, quer ela tenha sido implantada em um estágio ou não. O arquivo de definição exportado não inclui [extensões do API Gateway](#).



```
aws apigatewayv2 export-api \
 --api-id api-id \
 --output-type JSON \
 --specification OAS30 \
 --no-include-extensions \
 latest-api-definition.json
```

Para obter mais informações, consulte [ExportAPI](#) na Referência de API do Amazon API Gateway versão 2.

Exportar uma definição do OpenAPI 3.0 usando o console do API Gateway

O procedimento a seguir mostra como exportar uma definição do OpenAPI de uma API HTTP.

Como exportar uma definição do OpenAPI 3.0 usando o console do API Gateway

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione uma API HTTP.
3. No painel de navegação principal, em Desenvolver, escolha Exportar.
4. Selecione uma das seguintes opções para exportar a API:

API Gateway > APIs > my-http-api (abcdef1234) > Export

## Export

**Export an OpenAPI 3 definition** [Info](#)

Download an OpenAPI 3 definition of your latest changes or a stage's configuration.

Source

\$default ▼

Extensions [Learn more](#) [↗](#)

☒ Include API Gateway extensions

Output format

☒ JSON

☐ YAML

[Download](#)

- a. Em Origem, selecione uma origem para a definição do OpenAPI 3.0. Você pode escolher um estágio para exportar ou exportar a configuração mais recente da API.
  - b. Ative a opção Incluir extensões do API Gateway para incluir [extensões do API Gateway](#).
  - c. Em Formato de saída, selecione um formato de saída.
5. Escolha Baixar.

## Publicar APIs HTTP para os clientes invocarem

Você pode usar estágios e nomes de domínio personalizados para publicar sua API para que os clientes chamem.

O estágio de uma API é uma referência lógica a um estado do ciclo de vida de sua API (por exemplo, dev, prod, beta ou v2). Cada estágio é uma referência nomeada a uma implantação da API e é disponibilizado para chamadas feitas por aplicativos cliente. É possível configurar diferentes integrações e configurações para cada estágio de uma API.

É possível usar nomes de domínio personalizados para fornecer um URL mais simples e intuitivo para que os clientes chamem sua API do que o URL padrão, `https://api-id.execute-api.region.amazonaws.com/stage`.

### Note

Para aumentar a segurança das APIs do API Gateway, o domínio `execute-api.region.amazonaws.com` é registrado na [Public Suffix List \(PSL\)](#). Para maior segurança, recomendamos que você use cookies com um prefixo `__Host-` se precisar definir cookies confidenciais no nome de domínio padrão para as APIs do API Gateway. Essa prática ajudará a defender seu domínio contra tentativas de falsificação de solicitação entre sites (CSRF). Para obter mais informações, consulte a página [Set-Cookie](#) na Mozilla Developer Network.

### Tópicos

- [Estágios para APIs HTTP no API Gateway](#)
- [Política de segurança para APIs HTTP no API Gateway](#)
- [Nomes de domínio personalizados para APIs HTTP no API Gateway](#)

## Estágios para APIs HTTP no API Gateway

O estágio de uma API é uma referência lógica a um estado do ciclo de vida de sua API (por exemplo, dev, prod, beta ou v2). Os estágios de API são identificados por seu ID de API e nome de estágio e são incluídos no URL que você usa para chamar a API. Cada estágio é uma referência nomeada a uma implantação da API e é disponibilizado para chamadas feitas por aplicativos cliente.

É possível criar um estágio `$default` que é servido a partir da base do URL da sua API, por exemplo, `https://{api_id}.execute-api.{region}.amazonaws.com/`. Use esse URL para chamar um estágio de API.

Uma implantação é um instantâneo da configuração da API. Depois de implantar uma API em um estágio, ela estará disponível para ser chamada por clientes. Você deve implantar uma API para que as alterações entrem em vigor. Se você habilitar implantações automáticas, as alterações em uma API serão liberadas automaticamente para você.

### Usar variáveis de estágio para APIs HTTP no API Gateway

As variáveis de estágio são pares chave/valor que você pode definir para um estágio de uma API HTTP. Elas atuam como variáveis de ambiente e podem ser usadas na configuração da API.

As variáveis de estágio não se destinam a ser usadas para dados confidenciais, como credenciais. Para transmitir dados confidenciais para integrações, use um autorizador do AWS Lambda. Você pode passar dados confidenciais para integrações na saída do autorizador do Lambda. Para saber mais, consulte [the section called “Formato de resposta do autorizador do Lambda”](#).

Exemplo: usar uma variável de estágio para personalizar o endpoint de integração HTTP

Por exemplo, você pode definir uma variável de estágio e, depois, definir seu valor como um endpoint HTTP para uma integração de proxy HTTP. Posteriormente, você pode fazer referência ao endpoint usando o nome da variável de estágio associada. Fazendo isso, você pode usar a mesma configuração de API com um endpoint diferente em cada estágio. Da mesma forma, você pode usar variáveis de estágio para especificar uma integração de função diferente do AWS Lambda para cada estágio da API.

Para usar uma variável de estágio para personalizar o endpoint de integração HTTP, primeiro defina o nome e o valor da variável de estágio (por exemplo, `url`) com um valor de `example.com`. Depois, configure uma integração de proxy HTTP. Em vez de inserir o URL do endpoint, você pode instruir o API Gateway a usar o valor da variável de estágio, `http://{stageVariables.url}`. Esse valor

instrui o API Gateway a substituir sua variável de estágio `${}` em tempo de execução, dependendo do estágio da API.

É possível fazer referência a variáveis de estágio de forma semelhante para especificar um nome de função do Lambda ou um ARN de função da AWS.

Ao especificar um nome de função do Lambda como um valor de variável de estágio, você deve configurar as permissões nessa função do Lambda manualmente. O comando [add-permission](#) indicado abaixo configura a permissão para a função do 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
```

## Referência de variáveis de estágios do API Gateway para APIs HTTP no API Gateway

É possível usar variáveis de estágios do API Gateway para APIs HTTP nos casos a seguir.

### URIs de integração HTTP

Uma variável de estágio pode ser usada como parte de um URI de integração HTTP, como mostram os exemplos a seguir.

- Um URI completo sem protocolo – `http://${stageVariables.<variable_name>}`
- Um domínio complet – `http://${stageVariables.<variable_name>}/resource/operation`
- Um subdomíni – `http://${stageVariables.<variable_name>}.example.com/resource/operation`
- Um caminh – `http://example.com/${stageVariables.<variable_name>}/bar`
- Uma string de consult – `http://example.com/foo?q=${stageVariables.<variable_name>}`

### Funções do Lambda

É possível usar uma variável de estágio no lugar de um nome ou alias de integração da função do Lambda, conforme mostrado nos exemplos a seguir.

- `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 usar uma variável de estágio para uma função do Lambda, a função deve estar na mesma conta que a API. As variáveis de estágio não suportam funções do Lambda entre contas.

AWSCredenciais de integração da

É possível usar uma variável de estágio como parte de um ARN de credencial de usuário ou de função da AWS, conforme mostrado no exemplo a seguir.

- `arn:aws:iam::<account_id>:${stageVariables.<variable_name>}`

## Política de segurança para APIs HTTP no API Gateway

O API Gateway impõe uma política de segurança de TLS\_1\_2 para todos os endpoints da API HTTP.

Uma política de segurança é uma combinação predefinida da versão mínima do TLS e dos pacotes de criptografia oferecida pelo Amazon API Gateway. O protocolo TLS trata problemas de segurança de rede, como violação e interceptação entre um cliente e o servidor. Quando seus clientes estabelecem um handshake do TLS para a API por meio do domínio personalizado, a política de segurança aplica as opções do pacote de criptografia e da versão do TLS que seus clientes podem optar por usar. Essa política de segurança é compatível com tráfego TLS 1.2 e TLS 1.3 e rejeita tráfego TLS 1.0.

## Protocolos e cifras TLS compatíveis com APIs HTTP

A tabela a seguir descreve os protocolos TLS compatíveis com APIs HTTP.

| Protocolos TLS | Política de segurança TLS_1_2 |
|----------------|-------------------------------|
| TLSv1.3        | Sim                           |
| TLSv1.2        | Sim                           |

A tabela a seguir descreve as cifras TLS disponíveis para a política de segurança TLS 1\_2 para APIs HTTP.

| Cifras TLS                    | Política de segurança TLS_1_2 |
|-------------------------------|-------------------------------|
| TLS-AES-128-GCM-SHA256        | Sim                           |
| TLS-AES-256-GCM-SHA384        | Sim                           |
| TLS-CHACHA20-POLY1305-SHA256  | Sim                           |
| ECDHE-ECDSA-AES128-GCM-SHA256 | Sim                           |
| ECDHE-RSA-AES128-GCM-SHA256   | Sim                           |
| ECDHE-ECDSA-AES128-SHA256     | Sim                           |
| ECDHE-RSA-AES128-SHA256       | Sim                           |
| ECDHE-ECDSA-AES256-GCM-SHA384 | Sim                           |
| ECDHE-RSA-AES256-GCM-SHA384   | Sim                           |
| ECDHE-ECDSA-AES256-SHA384     | Sim                           |
| ECDHE-RSA-AES256-SHA384       | Sim                           |
| AES128-GCM-SHA256             | Sim                           |
| AES128-SHA256                 | Sim                           |
| AES256-GCM-SHA384             | Sim                           |

| Cifras TLS    | Política de segurança TLS_1_2 |
|---------------|-------------------------------|
| AES256-SHA256 | Sim                           |

## Nomes das criptografias OpenSSL e RFC

OpenSSL e IETF RFC 5246 usam nomes diferentes para as mesmas cifras. Para ver uma lista dos nomes das cifras, consulte [the section called “Nomes das criptografias OpenSSL e RFC”](#).

## Informações sobre APIs REST e APIs de WebSocket

Para saber mais sobre APIs REST e APIs de WebSocket, consulte [the section called “Escolher uma política de segurança”](#) e [the section called “Política de segurança para APIs de WebSocket no API Gateway”](#).

## Nomes de domínio personalizados para APIs HTTP no API Gateway

Os nomes de domínio personalizados são URLs mais simples e intuitivos que você pode fornecer aos usuários da API.

Após a implantação da sua API, você e seus clientes podem invocar essa API usando a URL de base padrão com o seguinte formato:

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

em que *api-id* é gerado pelo API Gateway, *region* é a região da AWS e *stage* é especificado por você ao implantar a API.

A parte do nome de host do URL, *api-id*.execute-api.*region*.amazonaws.com, refere-se a um endpoint de API. O endpoint de API padrão é gerado aleatoriamente, difícil de lembrar e não é simples de usar.

Com nomes de domínio personalizados, você pode configurar o nome de host da API e escolher um caminho base (por exemplo, myservice) para mapear o URL alternativo para sua API. Por exemplo, um URL de base de API mais amigável pode se tornar:

```
https://api.example.com/myservice
```

## Considerações

As considerações a seguir podem afetar o uso de um nome de domínio personalizado.

- Um nome de domínio regional personalizado pode ser associado a APIs REST e APIs HTTP. É possível usar as APIs do API Gateway versão 2 para criar e gerenciar nomes de domínio regionais personalizados para APIs REST.
- Para a versão mínima de TLS, somente o TLS 1.2 é compatível.
- É necessário criar ou atualizar o registro de recursos do provedor DNS para ser mapeado ao endpoint da API. Sem esse mapeamento, as solicitações de API que forem direcionadas para o nome de domínio personalizado não conseguirão acessar o API Gateway.
- É possível comportar um número quase infinito de nomes de domínio sem exceder a cota padrão usando um certificado curinga. Para obter mais informações, consulte [the section called “Nomes de domínio personalizados curinga”](#).

## Pré-requisitos

Veja a seguir os pré-requisitos para criar um nome de domínio personalizado.

### Registrar um nome de domínio

É necessário ter um nome de domínio da Internet registrado para configurar nomes de domínio personalizados para as APIs. É possível registrar o domínio da internet usando o [Amazon Route 53](#) ou um registrador de domínios de terceiros da sua escolha. O nome de domínio personalizado pode ser o nome de um subdomínio ou do domínio raiz (também conhecido como “ápex da zona”) de um domínio da internet registrado.

O nome de domínio deve seguir a especificação [RFC 1035](#) e pode ter no máximo 63 octetos por etiqueta e 255 octetos no total.

### Certificados para nomes de domínio personalizados

Antes de configurar um nome de domínio personalizado para uma API, você deve ter um certificado SSL/TLS pronto no ACM. Se o ACM não estiver disponível na região da AWS onde você está criando o nome de domínio personalizado, será necessário importar um certificado para o API Gateway nessa região.

Para importar um certificado SSL/TLS, você deve fornecer o corpo do certificado SSL/TLS formatado em PEM, sua chave privada e a cadeia de certificado para o nome de domínio personalizado.



Cada certificado armazenado no ACM é identificado por seu ARN. Com certificados emitidos pelo ACM, não é necessário se preocupar em expor detalhes de certificados confidenciais, como a chave privada. Para usar um certificado gerenciado pela AWS para um nome de domínio, basta fazer referência ao seu ARN.

Se o seu aplicativo usa a fixação de certificados, às vezes chamada de fixação SSL, para fixar um certificado do ACM, talvez o aplicativo não consiga se conectar ao seu domínio após a AWS renovar o certificado. Para ter mais informações, consulte [Problemas de fixação do certificado](#) no Guia do usuário do AWS Certificate Manager.

## Nomes de domínio personalizados curinga

Com nomes de domínio personalizados curinga, você pode suportar um número quase infinito de nomes de domínio sem exceder a [cota padrão](#). Por exemplo, você pode dar a cada um de seus clientes seu próprio nome de domínio `customername.api.example.com`.

Para criar um nome de domínio personalizado curinga, especifique um curinga (\*) como o primeiro subdomínio de um domínio personalizado que representa todos os subdomínios possíveis de um domínio raiz.

Por exemplo, o nome de domínio personalizado curinga `*.example.com` resulta em subdomínios, como `a.example.com`, `b.example.com` e `c.example.com`, que são todos roteados para o mesmo domínio.

Os nomes de domínio personalizados curinga oferecem suporte a configurações distintas dos nomes de domínio personalizados padrão do API Gateway. Por exemplo, em uma única conta da AWS, é possível configurar `*.example.com` e `a.example.com` para se comportarem de forma diferente.

Para criar um nome de domínio personalizado curinga, é necessário fornecer um certificado emitido pelo ACM que foi validado usando o DNS ou o método de validação por e-mail.

### Note

Não é possível criar um nome de domínio personalizado curinga se uma conta da AWS diferente tiver criado um nome de domínio personalizado que esteja em conflito com o nome de domínio personalizado curinga. Por exemplo, se a conta A tiver criado `a.example.com`, a conta B não poderá criar o nome de domínio personalizado curinga `*.example.com`. Se a conta A e a conta B compartilham um proprietário, entre em contato com a [Central de Suporte da AWS](#) para solicitar uma exceção.

## Próximas etapas para nomes de domínio personalizados

Para configurar um nome de domínio personalizado para uma API HTTP, use a documentação da seção API REST do Guia do desenvolvedor do API Gateway.

Primeiro, especifique o certificado para o nome de domínio personalizado. Para obter mais informações, consulte [the section called “Preparar certificados no AWS Certificate Manager”](#). Depois, crie um nome de domínio regional personalizado. Para obter mais informações, consulte [the section called “Configurar um nome de domínio regional personalizado”](#).

## Mapear estágios de API para um nome de domínio personalizado para APIs HTTP

Você usa mapeamentos de API para conectar estágios de API a um nome de domínio personalizado. Depois de criar um nome de domínio e configurar registros DNS, você usa mapeamentos de API para enviar tráfego para as suas APIs utilizando o seu nome de domínio personalizado.

Um mapeamento de API especifica uma API, um estágio e, opcionalmente, um caminho a usar para o mapeamento. Por exemplo, você pode mapear o estágio `production` de uma API para `https://api.example.com/orders`.

Você pode mapear os estágios da API HTTP e REST para o mesmo nome de domínio personalizado.

Antes de criar um mapeamento de API, você deve ter uma API, um estágio e um nome de domínio personalizado. Para saber mais sobre como criar um nome de domínio personalizado, consulte [the section called “Configurar um nome de domínio regional personalizado”](#).

## Rotear solicitações de API

Você pode configurar mapeamentos de API com vários níveis, por exemplo, `orders/v1/items` e `orders/v2/items`.

Para mapeamentos de API com vários níveis, o API Gateway encaminha as solicitações ao mapeamento de API que tem o prefixo correspondente mais longo. Para selecionar a API para invocar, o API Gateway considera apenas os caminhos configurados para mapeamentos de API, e não rotas de API. Se nenhum caminho corresponder à solicitação, o API Gateway enviará a solicitação para a API que você mapeou para o caminho vazio (`none`).

Para nomes de domínio personalizados que usam mapeamentos de API com vários níveis, o API Gateway encaminha as solicitações ao mapeamento de API que tem o prefixo correspondente mais longo.

Por exemplo, considere um nome de domínio personalizado `https://api.example.com` com os seguintes mapeamentos de API:

1. (none) mapeado para a API 1.
2. `orders` mapeado para a API 2.
3. `orders/v1/items` mapeado para a API 3.
4. `orders/v2/items` mapeado para a API 4.
5. `orders/v2/items/categories` mapeado para a API 5.

| Solicitação                                                       | API selecionada | Explicação                                                                                                                     |
|-------------------------------------------------------------------|-----------------|--------------------------------------------------------------------------------------------------------------------------------|
| <code>https://api.example.com/orders</code>                       | API 2           | A solicitação apresenta correspondência exata a esse mapeamento de API.                                                        |
| <code>https://api.example.com/orders/v1/items</code>              | API 3           | A solicitação apresenta correspondência exata a esse mapeamento de API.                                                        |
| <code>https://api.example.com/orders/v2/items</code>              | API 4           | A solicitação apresenta correspondência exata a esse mapeamento de API.                                                        |
| <code>https://api.example.com/orders/v1/items/123</code>          | API 3           | O API Gateway escolhe o mapeamento com o caminho correspondente mais longo. O 123 no final da solicitação não afeta a seleção. |
| <code>https://api.example.com/orders/v2/items/categories/5</code> | API 5           | O API Gateway escolhe o mapeamento com o caminho correspondente mais longo.                                                    |
| <code>https://api.example.com/customers</code>                    | API 1           | O API Gateway usa o mapeamento vazio como um catch-all.                                                                        |

| Solicitação                                        | API selecionada | Explicação                                                                                                                                                                                                                                                                                                                                                                            |
|----------------------------------------------------|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>https://api.example.com/ordersandmore</code> | API 2           | <p>O API Gateway escolhe o mapeamento com o prefixo correspondente mais longo.</p> <p>Para um nome de domínio personalizado configurado com mapeamentos de nível único, como somente <code>https://api.example.com/orders</code> e <code>https://api.example.com/</code>, o API Gateway escolheria a API 1, pois não há um caminho correspondente com <code>ordersandmore</code>.</p> |

## Restrições

- Em um mapeamento de API, o nome de domínio personalizado e as APIs mapeadas devem estar na mesma conta da AWS.
- Os mapeamentos de API devem conter apenas letras, números e os caracteres a seguir: `$-_.+!*'()/`.
- O comprimento máximo para o caminho em um mapeamento de API é de 300 caracteres.
- É possível ter 200 mapeamentos de API com vários níveis para cada nome de domínio. Esse limite não inclui mapeamento de API com um único nível, como `/prod`.
- Você só pode mapear APIs HTTP para um nome de domínio personalizado regional com a política de segurança TLS 1.2.
- Você não pode mapear APIs WebSocket para o mesmo nome de domínio personalizado que uma API HTTP ou API REST.
- Se você criar mapeamentos de API com vários níveis, o API Gateway mudará todos os nomes de cabeçalho para minúsculas.

## Crie um mapeamento de API

Para criar um mapeamento de API, você deve primeiro criar um nome de domínio personalizado, uma API e um estágio. Para obter informações sobre como criar um nome de domínio personalizado, consulte [the section called “Configurar um nome de domínio regional personalizado”](#).

Para obter exemplos de modelos do AWS Serverless Application Model que criam todos os recursos, consulte [Sessões com SAM](#) no GitHub.

## Console de gerenciamento da AWS

Para criar um mapeamento de API

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha Custom domain names (Nomes de domínios personalizados).
3. Selecione um nome de domínio personalizado que você já criou.
4. Escolha API mappings (Mapeamentos de API).
5. Escolha Configure API mappings (Configurar mapeamentos de API).
6. Escolha Add new mapping (Adicionar novo mapeamento).
7. Insira uma API, um Estágio e, opcionalmente, um Caminho.
8. Escolha Salvar.

## AWS CLI

O comando [create-api-mapping](#) a seguir cria um mapeamento de API. Neste exemplo, o API Gateway envia solicitações para `api.example.com/v1/orders` para a API e o estágio especificados.

```
aws apigatewayv2 create-api-mapping \
 --domain-name api.example.com \
 --api-mapping-key v1/orders \
 --api-id a1b2c3d4 \
 --stage test
```

## CloudFormation

O exemplo de CloudFormation a seguir cria um mapeamento de API.

```
MyApiMapping:
 Type: 'AWS::ApiGatewayV2::ApiMapping'
 Properties:
 DomainName: api.example.com
 ApiMappingKey: 'orders/v2/items'
 ApiId: !Ref MyApi
 Stage: !Ref MyStage
```

## Desabilitar o endpoint padrão para APIs HTTP

Por padrão, os clientes podem invocar sua API usando o endpoint `execute-api` gerado pelo API Gateway para sua API. Para garantir que os clientes possam acessar sua API somente usando um nome de domínio personalizado, desabilite o endpoint `execute-api` padrão. Quando o endpoint padrão é desabilitado, ele afeta todos os estágios de uma API.

O procedimento a seguir mostra como desabilitar o endpoint padrão de uma API HTTP.

### Console de gerenciamento da AWS

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione uma API HTTP.
3. Escolha o ID da API para abrir a página Detalhes da API.
4. Em Detalhes da API, escolha Editar.
5. Em Endpoint padrão, selecione Desabilitar.
6. Escolha Salvar.

Se você ativar as implantações automáticas no estágio, não será necessário reimplantar a API para que a alteração entre em vigor. Caso contrário, você deverá reimplantar a API.

7. (Opcional) Selecione Implantar e reimplante a API ou crie uma etapa para que a alteração entre em vigor.

### AWS CLI

O comando [update-domain-name](#) indicado abaixo desabilita o endpoint padrão para uma API HTTP:

```
aws apigatewayv2 update-api \
 --api-id abcdef123 \
 --disable-execute-api-endpoint
```

Depois de desabilitar o endpoint padrão, é necessário implantar sua API para que a alteração entre em vigor, a menos que as implantações automáticas estejam habilitadas.

O seguinte comando [create-deployment](#) cria uma implantação:

```
aws apigatewayv2 create-deployment \
 --api-id abcdef123 \
 --stage-name dev
```

## Tipos de endereço IP para nomes de domínio personalizados para APIs HTTP

Ao criar uma API, você especifica o tipo de endereços IP que podem invocar seu domínio. É possível escolher IPv4 para resolver endereços IPv4 e invocar seu domínio ou escolher pilha dupla para permitir que endereços IPv4 e IPv6 invoquem seu domínio. Recomendamos que você defina o tipo de endereço IP como pilha dupla para aliviar o esgotamento do espaço IP ou para seu procedimento de segurança. Para ter mais informações sobre os benefícios de um tipo de endereço IP de pilha dupla, consulte [IPv6 na AWS](#).

### Considerações sobre tipos de endereço IP

As considerações a seguir podem afetar o uso de tipos de endereço IP.

- O tipo de endereço IP padrão para nomes de domínio personalizados do API Gateway é IPv4.
- Seu nome de domínio personalizado não precisa ter o mesmo tipo de endereço IP para todas as APIs associadas a ele. Se você desabilitar seu endpoint de API padrão, isso poderá afetar como os chamadores podem invocar sua API.

### Alterar o tipo de endereço IP de um nome de domínio personalizado

É possível alterar o tipo de endereço IP atualizando a configuração do endpoint do domínio. É possível atualizar a configuração do endpoint do domínio usando o Console de gerenciamento da AWS, a AWS CLI, o CloudFormation ou um SDK da AWS.

## Console de gerenciamento da AWS

Como alterar o tipo de endereço IP de um nome de domínio personalizado

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha um nome de domínio personalizado público.
3. Escolha Configurações de endpoint.
4. Em Tipo de endereço IP, escolha IPv4 ou Pilha dupla.
5. Escolha Salvar.

## AWS CLI

O comando [update-domain-name](#) a seguir atualiza uma API para ter um tipo de endereço IP de pilha dupla:

```
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
```

A saída será exibida da seguinte forma:

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



# Proteger as APIs HTTP no API Gateway

O API Gateway fornece várias maneiras de proteger sua API de determinadas ameaças, como usuários mal-intencionados ou picos de tráfego. É possível proteger a sua API com estratégias como configurar alvos de controle de utilização e habilitar o TLS mútuo. Nesta seção, você pode aprender a habilitar esses recursos usando o API Gateway.

## Tópicos

- [Controle de utilização das solicitações das APIs HTTP para ter um melhor throughput no API Gateway](#)
- [Como ativar a autenticação do TLS mútuo para APIs HTTP no API Gateway](#)

## Controle de utilização das solicitações das APIs HTTP para ter um melhor throughput no API Gateway

Você pode configurar o controle de utilização para as suas APIs para ajudar a protegê-las da sobrecarga de numerosas solicitações. Os controles de utilização são aplicados de acordo com o melhor esforço e devem ser considerados alvos, e não limites máximos garantidos de solicitações.

O API Gateway controla a utilização das solicitações para a sua API usando o algoritmo do bucket de token, em que um token equivale a uma solicitação. Especificamente, o API Gateway analisa a taxa e uma intermitência de envios de solicitações de todas as APIs na sua conta, por região. No algoritmo do bucket de token, uma intermitência pode permitir a saturação predefinida desses limites, mas há alguns casos em que outros fatores também podem exceder tais limites.

Quando os envios de solicitações excederem a taxa de solicitação de estado fixo e os limites de intermitência, o API Gateway iniciará o controle de utilização de solicitações. Neste momento, pode ser que os clientes recebam respostas de erro 429 Too Many Requests. Ao capturar essas exceções, o cliente poderá reenviar as solicitações com falha de uma forma que restrinja as taxas.

Como desenvolvedor de APIs, você pode definir os limites alvo para estágios ou rotas de APIs particulares para melhorar a performance geral em todas as APIs na sua conta.

## Controle de utilização no nível da conta por região

Por padrão, o API Gateway controla a utilização das solicitações de estado fixo por segundo (RPS) em todas as APIs de uma conta da AWS, por região. Ele também limita a intermitência (ou seja, o tamanho máximo do bucket) em todas as APIs de uma conta da AWS, por região. No API Gateway,

o limite de intermitência representa o número máximo alvo de envios simultâneos de solicitações que ele fará antes de retornar respostas de erro 429 Too Many Requests. Para obter mais informações sobre cotas de controle de utilização, consulte [Cotas e observações importantes](#).

Os limites por conta são aplicados a todas as APIs em uma conta em uma região especificada. O limite de taxas no nível da conta pode ser aumentado por meio de uma solicitação; é possível obter limites mais altos com APIs com tempos limite mais curtos e cargas úteis menores. Para solicitar um aumento nos limites de controle de utilização no nível da conta por região, entre em contato com a [Central de Suporte da AWS](#). Para obter mais informações, consulte [Cotas e observações importantes](#). Observe que tais limites não podem ser superiores aos limites de controle de utilização da AWS.

## Limitação em nível de rota

Você pode definir a limitação no nível da rota para substituir as limitações de solicitação no nível da conta para um estágio específico ou para rotas individuais em sua API. Os limites de controle de utilização da rota padrão não podem exceder os limites de taxa em nível de conta.

É possível configurar o controle de utilização de nível de rota usando a AWS CLI. O comando [update-stage](#) indicado abaixo configura o controle de utilização personalizado para o estágio especificado e a rota de uma API:

```
aws apigatewayv2 update-stage \
 --api-id a1b2c3d4 \
 --stage-name dev \
 --route-settings '{"GET /pets":
{"ThrottlingBurstLimit":100, "ThrottlingRateLimit":2000}}'
```

## Como ativar a autenticação do TLS mútuo para APIs HTTP no API Gateway

A autenticação TLS mútua requer autenticação bidirecional entre o cliente e o servidor. Com TLS mútuo, os clientes devem apresentar certificados X.509 para verificar sua identidade a fim de acessar sua API. O TLS mútuo é um requisito comum para a Internet das Coisas (IoT) e aplicações business-to-business.

É possível usar o TLS mútuo juntamente com outras [operações de autorização e autenticação](#) compatíveis com o API Gateway. O API Gateway encaminha os certificados que os clientes fornecem aos autorizadores do Lambda e às integrações de backend.

### Important

Por padrão, os clientes podem invocar sua API usando o endpoint `execute-api` gerado pelo API Gateway para sua API. Para garantir que os clientes possam acessar sua API somente usando um nome de domínio personalizado com TLS mútuo, desabilite o endpoint `execute-api` padrão. Para saber mais, consulte [the section called “Desabilitar o endpoint padrão”](#).

## Pré-requisitos do TLS mútuo

Para configurar o TLS mútuo, você precisa de:

- Um nome de domínio personalizado
- Pelo menos um certificado configurado no AWS Certificate Manager para o seu nome de domínio personalizado
- Um armazenamento de confiança configurado e carregado no Amazon S3

### Nomes de domínios personalizados

Para habilitar o TLS mútuo para uma API HTTP, é necessário configurar um nome de domínio personalizado para sua API. Você pode habilitar o TLS mútuo para um nome de domínio personalizado e, depois, fornecer o nome de domínio personalizado aos clientes. Para acessar uma API usando um nome de domínio personalizado que tenha TLS mútuo habilitado, os clientes devem apresentar certificados confiáveis em solicitações de API. Você pode encontrar essas informações em [the section called “Nomes de domínios personalizados”](#).

### Uso de certificados emitidos pelo AWS Certificate Manager

É possível solicitar um certificado publicamente confiável diretamente do ACM ou importar certificados públicos ou autoassinados. Para configurar um certificado no ACM, acesse o [ACM](#). Para importar um certificado, continue lendo na seção a seguir.

### Usar um certificado importado ou Autoridade de Certificação Privada da AWS

Para usar um certificado importado para o ACM ou um certificado da Autoridade de Certificação Privada da AWS com TLS mútuo, o API Gateway precisa de um `ownershipVerificationCertificate` emitido pela ACM. Esse certificado de propriedade é utilizado apenas para verificar você se tem permissões para utilizar o

nome de domínio. Ele não é usado para o handshake TLS. Se você ainda não tem um `ownershipVerificationCertificate`, acesse <https://console.aws.amazon.com/acm/> para configurar um.

Você precisará manter esse certificado válido durante todo o tempo de vida do seu nome de domínio. Se um certificado expirar e a renovação automática falhar, todas as atualizações do nome de domínio serão bloqueadas. Você precisará atualizar o `ownershipVerificationCertificateArn` com um `ownershipVerificationCertificate` válido antes de poder fazer outras alterações. O `ownershipVerificationCertificate` não pode ser usado como um certificado de servidor para outro domínio de TLS mútuo no API Gateway. Se um certificado for reimportado diretamente para o ACM, o emissor deverá permanecer o mesmo.

### Configuração do armazenamento de confiança

Os armazenamentos de confiança são arquivos de texto com extensão `.pem`. Eles são uma lista confiável de certificados de autoridades de certificação. Para usar TLS mútuo, crie um armazenamento confiável de certificados X.509 que podem acessar sua API.

Você deve incluir a cadeia de confiança completa, começando pelo certificado da autoridade de certificação emissora até o certificado CA, em seu armazenamento de confiança. O API Gateway aceita certificados de cliente emitidos por qualquer autoridade de certificação presente na cadeia de confiança. Os certificados podem ser de autoridades de certificação públicas ou privadas. Eles podem ter um tamanho máximo de cadeia de quatro. Você também pode fornecer certificados autoassinados. Os seguintes algoritmos de hash são aceitos no armazenamento de confiança:

- SHA-256 ou mais forte
- RSA-2048 ou mais forte
- ECDSA-256 ou mais forte

O API Gateway valida várias propriedades de certificado. É possível usar autorizadores do Lambda para executar verificações adicionais quando um cliente invoca uma API, incluindo verificar se um certificado foi revogado. O API Gateway valida as seguintes propriedades:

| Validação     | Descrição                                                   |
|---------------|-------------------------------------------------------------|
| Sintaxe X.509 | O certificado deve atender aos requisitos da sintaxe X.509. |

| Validação                                    | Descrição                                                                                                                           |
|----------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| Integridade                                  | O conteúdo do certificado não pode ter sido alterado do assinado pela autoridade de certificação do armazenamento confiável.        |
| Validity                                     | O período de validade do certificado deve ser atual.                                                                                |
| Encadeamento de nomes/encadeamento de chaves | Os nomes e os assuntos dos certificados devem formar uma cadeia ininterrupta. Eles podem ter um tamanho máximo de cadeia de quatro. |

Fazer upload do armazenamento de confiança para um bucket do Amazon S3 em um único arquivo

Example certificates.pem

```
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Certificate contents>
-----END CERTIFICATE-----
...
```

O comando [cp](#) da AWS CLI indicado abaixo carrega `certificates.pem` em seu bucket do Amazon S3.

```
aws s3 cp certificates.pem s3://bucket-name
```

## Configurar o TLS mútuo para um nome de domínio personalizado

Para configurar o TLS mútuo para uma API HTTP, é necessário usar um nome de domínio personalizado regional para sua API, com uma versão do TLS mínima de 1.2. Para saber mais sobre

como criar e configurar um nome de domínio personalizado, consulte [the section called “Configurar um nome de domínio regional personalizado”](#).

 Note

O TLS mútuo não é suportado para APIs privadas.

Depois de fazer upload do armazenamento de confiança para o Amazon S3, você pode configurar o nome de domínio personalizado para usar TLS mútuo. O comando [create-domain-name](#) indicado abaixo cria um nome de domínio personalizado com TLS mútuo:

```
aws apigatewayv2 create-domain-name \
 --domain-name api.example.com \
 --domain-name-configurations CertificateArn=arn:aws:acm:us-west-2:123456789012:certificate/123456789012-1234-1234-12345678 \
 --mutual-tls-authentication TruststoreUri=s3://bucket-name/key-name
```

Depois de criar o nome de domínio, é necessário configurar registros DNS e mapeamentos de caminho base para operações da API. Para saber mais, consulte [Configurar um nome de domínio regional personalizado no API Gateway](#).

## Invocar uma API usando um nome de domínio personalizado que requer TLS mútuo

Para invocar uma API com TLS mútuo habilitado, os clientes precisam apresentar um certificado confiável na solicitação de API. Quando um cliente tenta invocar a API, o API Gateway procura o emissor do certificado de cliente no seu armazenamento de confiança. Para que o API Gateway prossiga com a solicitação, o emissor do certificado e a cadeia de confiança completa até o certificado CA raiz devem estar no seu armazenamento de confiança.

O comando `curl` demonstrativo a seguir envia uma solicitação para `api.example.com`, que inclui `my-cert.pem` na solicitação. `my-key.key` é a chave privada para o certificado.

```
curl -v --key ./my-key.key --cert ./my-cert.pem api.example.com
```

Sua API será invocada somente se o seu armazenamento de confiança confiar no certificado. As seguintes condições farão com que o API Gateway falhe no handshake TLS e negue a solicitação com um código de status 403. Se seu certificado:

- não é confiável
- expirou
- não usa um algoritmo compatível

#### Note

O API Gateway não verifica se um certificado foi revogado.

## Atualizar o armazenamento de confiança

Para atualizar os certificados no armazenamento de confiança, faça upload de um novo pacote de certificados para o Amazon S3. Em seguida, você poderá atualizar o nome de domínio personalizado para usar o certificado atualizado.

Use o [Versionamento do Amazon S3](#) para manter várias versões do armazenamento de confiança. Quando o nome de domínio personalizado é atualizado para usar uma nova versão de armazenamento de confiança, o API Gateway exibirá avisos se os certificados forem inválidos.

O API Gateway produz avisos de certificado somente quando o nome de domínio é atualizado. O API Gateway não o notificará se um certificado carregado anteriormente expirar.

O comando [update-domain-name](#) indicado abaixo atualiza um nome de domínio personalizado para usar uma nova versão do armazenamento de confiança:

```
aws apigatewayv2 update-domain-name \
 --domain-name api.example.com \
 --domain-name-configurations CertificateArn=arn:aws:acm:us-
west-2:123456789012:certificate/123456789012-1234-1234-12345678 \
 --mutual-tls-authentication TruststoreVersion='abcdef123'
```

## Desativar o TLS mútuo

Para desativar o TLS mútuo para um nome de domínio personalizado, remova o armazenamento de confiança do nome de domínio personalizado, conforme mostrado no comando a seguir.

O comando [update-domain-name](#) indicado abaixo atualiza um nome de domínio personalizado para remover o armazenamento de confiança do seu nome de domínio personalizado:

```
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=''
```

## Solucionar problemas de TLS mútuo para a API HTTP

O tópico a seguir fornece orientações para a solução de erros e problemas que você pode encontrar ao ativar o TLS mútuo.

### Solução de problemas de avisos de certificado

Ao criar um nome de domínio personalizado com TLS mútuo, o API Gateway exibirá avisos se os certificados no armazenamento de confiança não forem válidos. Isso também pode ocorrer ao atualizar um nome de domínio personalizado para usar um novo armazenamento de confiança. Os avisos indicam o problema com o certificado e o assunto que produziu o aviso. O TLS mútuo ainda está habilitado para sua API, mas alguns clientes podem não conseguir acessar a API.

Para identificar o certificado que produziu o aviso, é necessário decodificar os certificados em seu armazenamento de confiança. É possível usar ferramentas como `openssl` para decodificar os certificados e identificar os assuntos.

O comando a seguir exibe o conteúdo de um certificado, incluindo o assunto.

```
openssl x509 -in certificate.crt -text -noout
```

Atualize ou remova os certificados que produziram avisos e, depois, faça upload de um novo armazenamento de confiança para o Amazon S3. Após o upload do novo armazenamento de confiança, atualize o nome de domínio personalizado para usar o novo armazenamento de confiança.

### Solução de problemas de conflitos de nomes de domínios

O erro "The certificate subject <certSubject> conflicts with an existing certificate from a different issuer." significa que mais de uma autoridade de certificação emitiu um certificado para esse domínio. Para cada entidade no certificado, pode haver somente um emissor no API Gateway para domínios do TLS mútuos. Você precisará obter todos os seus certificados para esse assunto por meio de um único emissor. Se o problema for com um



certificado sobre o qual você não tem controle, e você é capaz de provar que é o dono do nome de domínio, [entre em contato com Suporte](#) para abrir um ticket.

### Solução de problemas de mensagens de status de nomes de domínio

PENDING\_CERTIFICATE\_REIMPORT: isso significa que você reimportou um certificado para o ACM e ele falhou na validação porque o novo certificado tem um SAN (nome alternativo de entidade) que não é coberto pelo ownershipVerificationCertificate ou o assunto ou os SANs no certificado não cobrem o nome de domínio. Algo pode estar configurado incorretamente ou um certificado inválido foi importado. Você precisa reimportar um certificado válido para o ACM. Para obter mais informações sobre a validação, consulte [Validação da propriedade de domínios](#).

PENDING\_OWNERSHIP\_VERIFICATION: isso significa que seu certificado verificado anteriormente expirou e o ACM não conseguiu renová-lo automaticamente. Você precisará renovar o certificado ou solicitar um novo certificado. Mais informações sobre renovação de certificados podem ser encontradas no guia [Solução de problemas de renovação de certificados gerenciados do ACM](#).

## Monitorar APIs HTTP no API Gateway

É possível usar métricas do CloudWatch e CloudWatch Logs para monitorar APIs HTTP. Combinando logs e métricas, você pode registrar erros em log e monitorar o desempenho da API.

### Note

O API Gateway pode não gerar logs e métricas nos seguintes casos:

- Erros 413 de entidade de solicitação muito grande
- Erros 429 de muitas solicitações excessivos
- Erros da série 400 de solicitações enviadas a um domínio personalizado que não tem mapeamento de API
- Erros da série 500 causados por falhas internas

### Tópicos

- [Monitorar métricas do CloudWatch para APIs HTTP no API Gateway](#)
- [Configurar registro em log para APIs HTTP no API Gateway](#)

## Monitorar métricas do CloudWatch para APIs HTTP no API Gateway

É possível monitorar a execução da API usando o CloudWatch, que coleta e processa dados brutos do API Gateway em métricas legíveis, quase em tempo real. Essas estatísticas são registradas para um período de 15 meses, de forma que você possa acessar informações históricas e ganhar uma perspectiva melhor sobre como seu serviço ou aplicativo web está se saindo. Por padrão, os dados de métricas do API Gateway são enviados automaticamente para o CloudWatch em períodos de um minuto. Para monitorar suas métricas, crie um painel do CloudWatch para sua API. Para ter mais informações sobre como criar um painel do CloudWatch, consulte [Criar um painel do CloudWatch](#) no Guia do usuário do Amazon CloudWatch. Para obter mais informações, consulte [O que é o Amazon CloudWatch?](#) no Guia do usuário do Amazon CloudWatch.

As métricas a seguir são compatíveis com APIs HTTP. Você também pode habilitar métricas detalhadas para gravar métricas de nível de rota no Amazon CloudWatch.

| Métrica            | Descrição                                                                                                                  |
|--------------------|----------------------------------------------------------------------------------------------------------------------------|
| 4xx                | O número de erros no lado do cliente capturados em um determinado período.                                                 |
| 5xx                | O número de erros do servidor capturados em um período determinado.                                                        |
| Contagem           | O número total de solicitações de API em um determinado período.                                                           |
| IntegrationLatency | O tempo que o API Gateway leva para receber uma resposta do backend depois de retransmitir uma solicitação para o backend. |
| Latência           | O tempo que o API Gateway leva para devolver uma resposta para um cliente                                                  |

| Métrica       | Descrição                                                                                                                         |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------|
|               | depois de receber uma solicitação do cliente. A latência inclui a latência de integração e outras despesas gerais do API Gateway. |
| DataProcessed | A quantidade de dados processados em bytes.                                                                                       |

É possível usar as dimensões na tabela a seguir para filtrar métricas do API Gateway.

| Dimensão                        | Descrição                                                                                                                                                                                                                                                                                                                                                     |
|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Apild                           | Filtra as métricas do API Gateway para uma API com o ID de API especificado.                                                                                                                                                                                                                                                                                  |
| ID da API, estágio              | Filtra métricas do API Gateway para um estágio de API com o ID da API especificada e o ID do estágio.                                                                                                                                                                                                                                                         |
| Apild, método, recurso, estágio | <p>Filtra métricas do API Gateway para um método de API com o ID da API especificada, ID do estágio, caminho do recurso e ID da rota.</p> <p>O API Gateway não enviará essas métricas a menos que você tenha habilitado explicitamente métricas detalhadas do CloudWatch. Isso pode ser feito ao chamar a ação <a href="#">UpdateStage</a> da API REST V2</p> |

| Dimensão | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|          | do API Gateway para atualizar a propriedade <code>detailedMetricsEnabled</code> como <code>true</code> . Como alternativa, você pode chamar o comando <a href="#">update-stage</a> da AWS CLI para atualizar a propriedade <code>DetailedMetricsEnabled</code> para <code>true</code> . Permitir essas métricas incorrerá em cobranças adicionais na conta. Para obter informações de definição de preço, consulte <a href="#">Definição de preço do Amazon CloudWatch</a> . |

## Configurar registro em log para APIs HTTP no API Gateway

É possível ativar o registro em log para gravar logs no CloudWatch Logs. Você pode usar [variáveis de registro em log](#) para personalizar o conteúdo de seus logs.

Para melhorar seu procedimento de segurança, recomendamos que você grave logs no CloudWatch Logs referentes a todos os estágios da sua API HTTP. Você pode precisar fazer isso para cumprir vários requisitos de conformidade. Para ter mais informações, consulte [Amazon API Gateway controls](#) no Guia do usuário do AWS Security Hub.

Para ativar o registro em log para uma API HTTP, é necessário fazer o seguinte.

1. Seu usuário deve ter as permissões necessárias para ativar o registro em log.
2. Crie um grupo de logs do CloudWatch Logs.
3. Forneça o ARN do grupo de logs do CloudWatch Logs para um estágio de sua API.

### Permissões para ativar o registro em log

Para ativar o registro em log para uma API, seu usuário deve ter as permissões a seguir.

## 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": "*"
 }
]
}
```

## Criar um grupo de logs e ativar o registro em log para APIs HTTP

É possível criar um grupo de logs e ativar o registro em log de acesso usando o Console de gerenciamento da AWS ou a AWS CLI.

## Console de gerenciamento da AWS

1. Crie um grupo de logs.

Para saber como criar um grupo de logs usando o console, consulte [Criar um grupo de logs no Guia do usuário do Amazon CloudWatch Logs](#).

2. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
3. Selecione uma API HTTP.
4. Na guia Monitor no painel de navegação principal, selecione Logging (Registro em log).
5. Selecione um estágio para ativar o registro em log e Select (Selecionar).
6. Selecione Edit (Editar) para ativar o registro em log de acesso.
7. Ative Access logging (Registro em log de acesso), insira um CloudWatch Logs e selecione um formato de log.
8. Escolha Salvar.

## AWS CLI

O comando [create-log-group](#) indicado abaixo cria um grupo de logs:

```
aws logs create-log-group --log-group-name my-log-group
```

Você precisa do nome do recurso da Amazon (ARN) do seu grupo de logs para ativar o registro em log. O formato do ARN é `arn:aws:logs:region:account-id:log-group:log-group-name`.

O comando [update-stage](#) indicado abaixo ativa o registro em log para o estágio `$default` de uma 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"}'
```

## Formatos de log demonstrativos

Alguns formatos demonstrativos de log de acesso usados com frequência estão disponíveis no console do API Gateway e são listados a seguir.

- CLF ([Formato de log comum](#)):

```
$context.identity.sourceIp - - [$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 vírgula):

```
$context.identity.sourceIp,$context.requestTime,$context.httpMethod,
$context.routeKey,$context.protocol,$context.status,$context.responseLength,
$context.requestId,$context.extendedRequestId
```

## Personalizar logs de acesso à API HTTP

É possível usar as variáveis a seguir para personalizar logs de acesso para APIs HTTP. Para saber mais sobre logs de acesso para APIs HTTP, consulte [Configurar registro em log para APIs HTTP no API Gateway](#).

| Parâmetro                                                     | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$context.accountId</code>                              | O ID da conta da AWS do proprietário da API                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <code>\$context.apiId</code>                                  | O identificador que o API Gateway atribui à sua API.                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <code>\$context.authorizer.claims.<br/><i>property</i></code> | <p>Uma propriedade das declarações retornada s do JSON Web Token (JWT) depois que o chamador do método é autenticado com êxito, como <code>\$context.authorizer.claims.username</code>. Para obter mais informações, consulte <a href="#">Controlar o acesso a APIs HTTP com autorizadores JWT no API Gateway</a>.</p> <div> <b>Note</b><br/>Chamar <code>\$context.authorizer.claims</code> retorna um valor nulo.</div> |
| <code>\$context.authorizer.error</code>                       | A mensagem de erro retornada de um autorizador.                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <code>\$context.authorizer.<br/><i>property</i></code>        | <p>O valor do par de chave/valor especificado do mapa <code>context</code> retornado por uma função de autorizador do Lambda do API Gateway. Por exemplo, se o autorizador retornar o seguinte mapa <code>context</code>:</p> <div><pre>"context" : {<br/>  "key": "value",<br/>  "numKey": 1,<br/>  "boolKey": true<br/>}</pre></div> <p>chamar <code>\$context.authorizer.key</code> retorna a string "value", chamar <code>\$context.authorizer.numKey</code> retorna 1 e chamar</p>                    |



| Parâmetro                                           | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-----------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                     | <code>\$context.authorizer.boolKey</code> retorna true.                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <code>\$context.awsEndpointRequestId</code>         | O ID da solicitação do endpoint da AWS do cabeçalho <code>x-amz-request-id</code> ou <code>x-amzn-requestId</code> .                                                                                                                                                                                                                                                                                                                                                                                  |
| <code>\$context.awsEndpointRequestId2</code>        | O ID da solicitação do endpoint da AWS do cabeçalho <code>x-amz-id-2</code> .                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>\$context.customDomain.basePathMatched</code> | O caminho para um mapeamento da API correspondente a uma solicitação de entrada. Aplicável quando um cliente usa um nome de domínio personalizado para acessar uma API. Por exemplo, se um cliente enviar uma solicitação para <code>https://api.example.com/v1/orders/1234</code> e a solicitação corresponder ao mapeamento da API com o caminho <code>v1/orders</code> , o valor será <code>v1/orders</code> . Para saber mais, consulte <a href="#">the section called “Mapeamentos de API”</a> . |
| <code>\$context.dataProcessed</code>                | A quantidade de dados processados em bytes.                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <code>\$context.domainName</code>                   | O nome de domínio completo usado para invocar a API. Ele deve ser o mesmo que o cabeçalho Host de entrada.                                                                                                                                                                                                                                                                                                                                                                                            |
| <code>\$context.domainPrefix</code>                 | O primeiro rótulo do <code>\$context.domainName</code> e .                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <code>\$context.error.message</code>                | Uma string que contém uma mensagem de erro do API Gateway.                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <code>\$context.error.messageString</code>          | O valor citado de <code>\$context.error.message</code> , ou seja, <code>"\$context.error.message"</code> .                                                                                                                                                                                                                                                                                                                                                                                            |

| Parâmetro                                 | Descrição                                                                                                                                                                                                                                  |
|-------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$context.error.responseType</code> | Um tipo de <code>GatewayResponse</code> . Para obter mais informações, consulte <a href="#">the section called “Métricas”</a> e <a href="#">the section called “Configurar respostas do gateway para personalizar respostas de erro”</a> . |
| <code>\$context.extendedRequestId</code>  | Equivale a <code>\$context.requestId</code> .                                                                                                                                                                                              |
| <code>\$context.httpMethod</code>         | O método HTTP utilizado. Os valores válidos incluem: DELETE, GET, HEAD, OPTIONS, PATCH, POST e PUT.                                                                                                                                        |
| <code>\$context.identity.accountId</code> | O ID da conta da AWS associado à solicitação. Compatível com rotas que usam a autorização do IAM.                                                                                                                                          |
| <code>\$context.identity.caller</code>    | O identificador da entidade do autor da chamada que assinou a solicitação. Compatível com rotas que usam a autorização do IAM.                                                                                                             |

| Parâmetro                                                     | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|---------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$context.identity.cognitoAuthenticationProvider</code> | <p>Uma lista separada por vírgulas de todos os provedores de autenticação do Amazon Cognito usados pelo autor da chamada que faz a solicitação. Disponível somente se a solicitação foi assinada com credenciais do Amazon Cognito.</p> <p>Por exemplo, para uma identidade e de um grupo de usuários do Amazon Cognito, <code>cognito-idp.<i>region</i>.amazonaws.com/<i>user_pool_id</i></code>, <code>cognito-idp.<i>region</i>.amazonaws.com/<i>user_pool_id</i></code>:CognitoSignIn: <i>token subject claim</i></p> <p>Consulte informações sobre os provedores de autenticação do Amazon Cognito disponíveis em <a href="#">Using Federated Identities</a> no Guia do desenvolvedor do Amazon Cognito.</p> |
| <code>\$context.identity.cognitoAuthenticationType</code>     | <p>O tipo de autenticação do Amazon Cognito do autor da chamada que faz a solicitação. Disponível somente se a solicitação foi assinada com credenciais do Amazon Cognito. Os valores possíveis incluem <code>authenticated</code> para identidades autenticadas e <code>unauthenticated</code> para identidades não autenticadas.</p>                                                                                                                                                                                                                                                                                                                                                                            |
| <code>\$context.identity.cognitoIdentityId</code>             | <p>O ID de identidade do Amazon Cognito do autor da chamada que faz a solicitação. Disponível somente se a solicitação foi assinada com credenciais do Amazon Cognito.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

| Parâmetro                                                            | Descrição                                                                                                                                                                                                                |
|----------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$context.identity.cognitoIdentityPoolId</code>                | O ID do grupo de identidades do Amazon Cognito do autor da chamada que faz a solicitação. Disponível somente se a solicitação foi assinada com credenciais do Amazon Cognito.                                            |
| <code>\$context.identity.principalOrgId</code>                       | O <a href="#">ID da organização da AWS</a> . Compatível com rotas que usam a autorização do IAM.                                                                                                                         |
| <code>\$context.identity.clientCertificate.clientCertPem</code>      | O certificado de cliente codificado por PEM que o cliente apresentou durante a autenticação TLS mútua. Presente quando um cliente acessa uma API usando um nome de domínio personalizado que tenha TLS mútuo habilitado. |
| <code>\$context.identity.clientCertificate.subjectDN</code>          | O nome distinto do assunto do certificado apresentado por um cliente. Presente quando um cliente acessa uma API usando um nome de domínio personalizado que tenha TLS mútuo habilitado.                                  |
| <code>\$context.identity.clientCertificate.issuerDN</code>           | O nome distinto do emissor do certificado apresentado por um cliente. Presente quando um cliente acessa uma API usando um nome de domínio personalizado que tenha TLS mútuo habilitado.                                  |
| <code>\$context.identity.clientCertificate.serialNumber</code>       | O número de série do certificado. Presente quando um cliente acessa uma API usando um nome de domínio personalizado que tenha TLS mútuo habilitado.                                                                      |
| <code>\$context.identity.clientCertificate.validity.notBefore</code> | A data antes da qual o certificado é inválido. Presente quando um cliente acessa uma API usando um nome de domínio personalizado que tenha TLS mútuo habilitado.                                                         |

| Parâmetro                                                           | Descrição                                                                                                                                                                                                                                                                                                                   |
|---------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$context.identity.clientCertificate.validity.notAfter</code> | A data após a qual o certificado é inválido. Presente quando um cliente acessa uma API usando um nome de domínio personalizado que tenha TLS mútuo habilitado.                                                                                                                                                              |
| <code>\$context.identity.sourceIp</code>                            | O endereço IP de origem da conexão TCP mais próxima que está fazendo a solicitação para o endpoint do API Gateway.                                                                                                                                                                                                          |
| <code>\$context.identity.user</code>                                | O identificador da entidade do usuário que será autorizado contra o acesso a recursos. Compatível com rotas que usam a autorização do IAM.                                                                                                                                                                                  |
| <code>\$context.identity.userAgent</code>                           | O cabeçalho <u>User-Agent</u> do autor da chamada da API.                                                                                                                                                                                                                                                                   |
| <code>\$context.identity.userArn</code>                             | O Nome do Recurso Amazon (ARN) do usuário efetivo identificado após a autenticação. Compatível com rotas que usam a autorização do IAM. Para obter mais informações, consulte <a href="https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html">https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html</a> . |
| <code>\$context.integration.error</code>                            | A mensagem de erro retornada de uma integração. Equivale a <code>\$context.integration.errorMessage</code> .                                                                                                                                                                                                                |
| <code>\$context.integration.integrationStatus</code>                | Para a integração de proxy do Lambda, o código de status retornado pelo AWS Lambda, e não pelo código de função do Lambda de backend.                                                                                                                                                                                       |
| <code>\$context.integration.latency</code>                          | A latência de integração em ms. Equivale a <code>\$context.integrationLatency</code> .                                                                                                                                                                                                                                      |

| Parâmetro                                      | Descrição                                                                                                                                                  |
|------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$context.integration.requestId</code>   | O ID da solicitação do endpoint da AWS. Equivale a <code>\$context.awsEndpointRequestId</code> .                                                           |
| <code>\$context.integration.status</code>      | O código de status retornado de uma integração. Para integrações de proxy do Lambda, esse é o código de status que seu código de função do Lambda retorna. |
| <code>\$context.integrationErrorMessage</code> | Uma string que contém uma mensagem de erro de integração.                                                                                                  |
| <code>\$context.integrationLatency</code>      | A latência de integração em ms.                                                                                                                            |
| <code>\$context.integrationStatus</code>       | Para a integração de proxy do Lambda, esse parâmetro representa o código de status retornado pelo AWS Lambda, e não pela função do Lambda de backend.      |
| <code>\$context.path</code>                    | O caminho da solicitação. Por exemplo, <code>/ {stage}/root/child</code> .                                                                                 |
| <code>\$context.protocol</code>                | O protocolo de solicitação, por exemplo, <code>HTTP/1.1</code> .                                                                                           |

 Note

As APIs do API Gateway podem aceitar solicitações HTTP/2, mas o API Gateway envia solicitações para integrações de back-end usando HTTP/1.1. Como resultado, o protocolo de solicitação é registrado como HTTP/1.1 mesmo se um cliente enviar uma solicitação que usa HTTP/2.

| Parâmetro                               | Descrição                                                                              |
|-----------------------------------------|----------------------------------------------------------------------------------------|
| <code>\$context.requestId</code>        | O ID que o API Gateway atribui à solicitação de API.                                   |
| <code>\$context.requestTime</code>      | O horário da solicitação <a href="#">CLF</a> formatado (dd/MMM/yyyy:HH:mm:ss +-hhmm ). |
| <code>\$context.requestTimeEpoch</code> | O horário da solicitação <a href="#">Epoch</a> formatado.                              |
| <code>\$context.responseLatency</code>  | A latência da resposta em ms.                                                          |
| <code>\$context.responseLength</code>   | O tamanho da carga de resposta em bytes.                                               |
| <code>\$context.routeKey</code>         | A chave de rota da solicitação da API, por exemplo, /pets.                             |
| <code>\$context.stage</code>            | O estágio de implantação da solicitação de API (por exemplo, beta ou prod).            |
| <code>\$context.status</code>           | O status de resposta do método.                                                        |

## Solução de problemas com APIs HTTP no API Gateway

Os tópicos a seguir fornecem orientações para a solução de erros e problemas que podem ser encontrados durante o uso de APIs HTTP.

### Tópicos

- [Solução de problemas com integrações do Lambda para APIs HTTP](#)
- [Solução de problemas com os autorizadores JWT da API HTTP](#)

## Solução de problemas com integrações do Lambda para APIs HTTP

O tópico a seguir fornece orientações para a solução de erros e problemas que podem ser encontrados durante o uso de [AWS LambdaIntegrações do](#) com APIs HTTP.

## Problema: minha API com uma integração do Lambda retorna **{"message": "Internal Server Error"}**

Para solucionar o erro interno do servidor, adicione a `$context.integrationErrorMessage` [variável de registro em log](#) ao formato de log e visualize os logs da API HTTP. Para conseguir isso, faça o seguinte:

Como criar um grupo de logs usando o Console de gerenciamento da AWS

1. Abra o console do CloudWatch em <https://console.aws.amazon.com/cloudwatch/>.
2. Escolha Grupos de logs.
3. Escolha Criar grupo de logs.
4. Insira um nome para o grupo de logs e escolha Criar.
5. Anote o nome de recurso da Amazon (ARN) do grupo de logs. O formato do ARN é `arn:aws:logs:region:account-id:log-group:log-group-name`. O ARN do grupo de logs é necessário para habilitar o registro de acesso em logs para a API HTTP.

Como adicionar a variável de registro em log `$context.integrationErrorMessage`

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha sua API HTTP.
3. Em Monitor, escolha Registro em log.
4. Selecione um estágio da API.
5. Escolha Editar e habilite o registro de acesso em logs.
6. Para o destino do log, insira o ARN do grupo de log que você criou na etapa anterior.
7. Em Formato de log, escolha CLF. O API Gateway cria um exemplo de formato de log.
8. Adicione `$context.integrationErrorMessage` ao final do formato de log.
9. Escolha Save (Salvar).

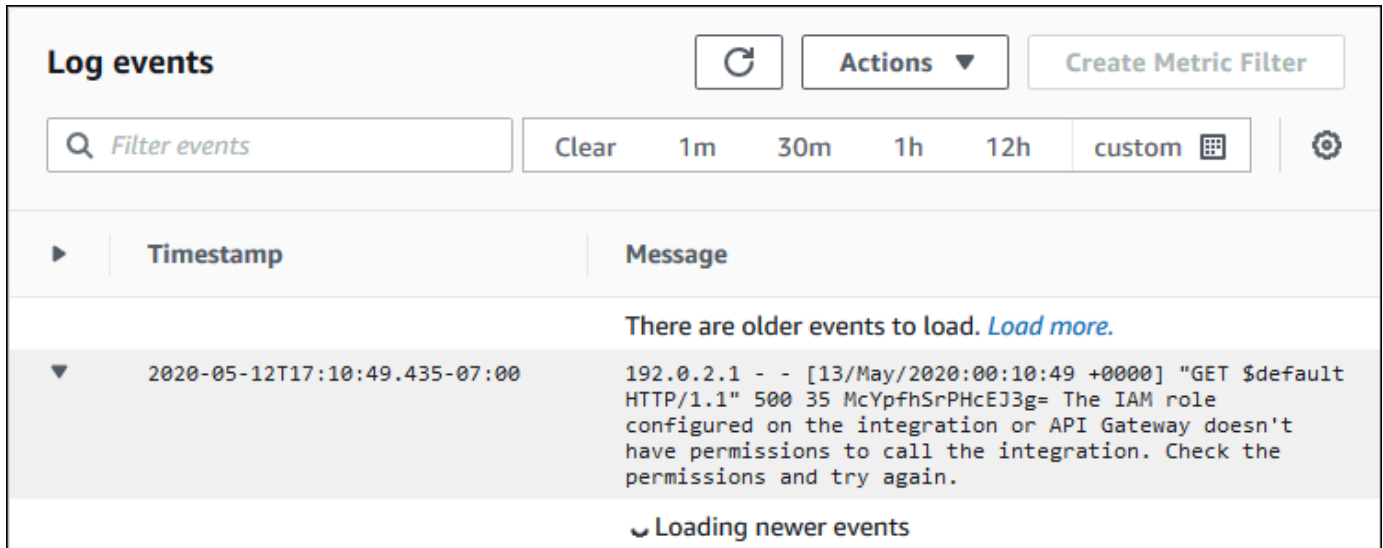
Como visualizar os logs da API

1. Gere os logs. Use um navegador ou `curl` para invocar a API.

```
$curl https://api-id.execute-api.us-west-2.amazonaws.com/route
```



2. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
3. Escolha sua API HTTP.
4. Em Monitor, escolha Registro em log.
5. Selecione o estágio da API para o qual você habilitou o registro em log.
6. Escolha View logs in CloudWatch (Exibir logs no CloudWatch).
7. Escolha o stream de logs mais recente para visualizar os logs da API HTTP.
8. A entrada de log deve ser semelhante à seguinte:



The screenshot shows the 'Log events' interface in the AWS API Gateway console. It includes a search bar with the placeholder 'Filter events', a 'Clear' button, and time range filters: '1m', '30m', '1h', '12h', and 'custom'. There is also a 'Create Metric Filter' button and a settings gear icon. The log events are displayed in a table with columns 'Timestamp' and 'Message'. A single log entry is shown with a timestamp of '2020-05-12T17:10:49.435-07:00' and a message indicating an integration error. Below the log entry, there is a 'Loading newer events' button.

| Timestamp                     | Message                                                                                                                                                                                                                                      |
|-------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2020-05-12T17:10:49.435-07:00 | 192.0.2.1 - - [13/May/2020:00:10:49 +0000] "GET \$default HTTP/1.1" 500 35 McYpfhSrPHcEJ3g= The IAM role configured on the integration or API Gateway doesn't have permissions to call the integration. Check the permissions and try again. |

Como adicionamos `$context.integrationErrorMessage` ao formato de log, vemos uma mensagem de erro nos logs que resume o problema.

Os logs podem incluir uma mensagem de erro diferente que indica que há um problema com o código de função do Lambda. Nesse caso, confira o código de função do Lambda e verifique se a função do Lambda retorna uma resposta no [formato necessário](#). Se os logs não incluírem uma mensagem de erro, adicione `$context.error.message` e `$context.error.responseType` ao seu formato de log para obter mais informações para ajudar a solucionar problemas.

Nesse caso, os logs mostram que o API Gateway não tinham as permissões necessárias para invocar a função do Lambda.

Quando você cria uma integração do Lambda no console do API Gateway, este configura automaticamente as permissões para invocar a função do Lambda. Ao criar uma integração do Lambda usando a AWS CLI, CloudFormation ou um SDK, você deve conceder permissões para

o API Gateway invocar a função. Os comandos [add-permission](#) indicados abaixo concedem permissões para possibilitar que diferentes rotas da API HTTP invoquem uma função do Lambda.

Example Exemplo: para o estágio **\$default** e a rota **\$default** de uma 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 Exemplo: para o estágio **prod** e a rota **test** de uma 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 a política de função](#) na guia Permissions (Permissões) do console do Lambda.

Tente invocar a API novamente. Você deverá ver a resposta da função do Lambda.

## Solução de problemas com os autorizadores JWT da API HTTP

O tópico a seguir fornece orientações para a solução de erros e problemas que podem ser encontrados durante o uso dos autorizadores JSON Web Token (JWT) com APIs HTTP.

Problema: minha API retorna **401 {"message":"Unauthorized"}**

Verifique o cabeçalho `www-authenticate` na resposta da API.

O comando a seguir usa `curl` para enviar uma solicitação para uma API com um autorizador de JWT que usa `$request.header.Authorization` como sua origem de identidade.

```
$curl -v -H "Authorization: token" https://api-id.execute-api.us-west-2.amazonaws.com/route
```

A resposta da API inclui um cabeçalho `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"}}
```

Nesse caso, o cabeçalho `www-authenticate` mostra que o token não foi emitido para um público válido. Para o API Gateway autorizar uma solicitação, a declaração `aud` ou `client_id` do JWT deve corresponder a uma das entradas de público configuradas para o autorizador. O API Gateway valida `client_id` somente se `aud` não estiver presente. Quando `aud` e `client_id` estão presentes, o API Gateway avalia `aud`.

Também é possível decodificar um JWT e verificar se ele corresponde ao emissor, ao público e aos escopos que a API exige. O site [jwt.io](https://jwt.io) pode depurar JWTs no navegador. A OpenID Foundation mantém uma [lista de bibliotecas para trabalhar com JWTs](#).

Para saber mais sobre os autorizadores JWT, consulte [Controlar o acesso a APIs HTTP com autorizadores JWT no API Gateway](#).

# APIs de WebSocket do API Gateway

Uma API WebSocket no API Gateway é uma coleção de rotas WebSocket integradas a endpoints HTTP de back-end, funções do Lambda ou outros serviços da AWS. É possível usar os recursos do API Gateway para obter ajuda em todos os aspectos do ciclo de vida da API, desde a criação até o monitoramento das APIs de produção.

As APIs WebSocket do API Gateway são bidirecionais. Um cliente pode enviar mensagens para um serviço, e os serviços podem enviar mensagens de forma independente para clientes. Esse comportamento bidirecional permite interações cliente/serviço mais ricas porque os serviços podem enviar dados por push para os clientes sem exigir que os clientes façam uma solicitação explícita. As APIs WebSocket são usadas com frequência em aplicativos em tempo real, como aplicativos de bate-papo, plataformas de colaboração, jogos multijogador e plataformas de negociação financeira.

Para obter um exemplo de aplicação para começar, consulte [Tutorial: Crie uma aplicação de bate-papo de WebSocket com uma API de WebSocket, o Lambda e o DynamoDB](#).

Nesta seção, é possível aprender a desenvolver, publicar, proteger e monitorar APIs WebSocket usando o API Gateway.

## Tópicos

- [Visão geral das APIs de WebSocket no API Gateway](#)
- [Desenvolver APIs de WebSocket no API Gateway](#)
- [Publicar APIs de WebSocket para os clientes invocarem](#)
- [Proteger as APIs de WebSocket no API Gateway](#)
- [Monitorar as APIs de WebSocket no API Gateway](#)

## Visão geral das APIs de WebSocket no API Gateway

No API Gateway, é possível criar uma API WebSocket como um frontend com estado para um serviço da AWS (como Lambda ou DynamoDB) ou para um endpoint HTTP. A API WebSocket invoca o backend com base no conteúdo de mensagens que recebe a partir de aplicativos do cliente.

Ao contrário de uma API REST, que recebe e responde a solicitações, uma API WebSocket oferece suporte a comunicações bidirecionais entre aplicativos do cliente e backend. O backend pode enviar mensagens de retorno para clientes conectados.

Na sua API WebSocket, mensagens JSON de entrada são direcionadas para integrações de backend com base nas rotas que você configurar. (Mensagens que não apresentam o formato JSON são direcionadas para a rota `$default` que você configurar).

Uma rota inclui uma chave de roteamento, que é o valor esperado quando uma expressão de seleção de rotas é avaliada. O `routeSelectionExpression` é um atributo definido em nível de API. Ele especifica uma propriedade JSON que deve estar presente na carga da mensagem. Para obter mais informações sobre expressões de seleção de rota, consulte [the section called “”](#).

Por exemplo, se as suas mensagens JSON contêm uma propriedade `action` e você deseja realizar ações diferentes de acordo com essa propriedade, sua expressão de seleção de rotas pode ser `${request.body.action}`. A tabela de roteamento deve especificar qual ação executar ao corresponder o valor da propriedade `action` com os valores de chave de rotas personalizada que você definiu na tabela.

## Usar rotas para uma API de WebSocket

Há três rotas predefinidas que podem ser usadas: `$connect`, `$disconnect` e `$default`. Além disso, você pode criar rotas personalizadas.

- O API Gateway chama a rota `$connect` quando uma conexão persistente entre o cliente e uma API WebSocket está sendo iniciada.
- O API Gateway chama a rota `$disconnect` quando o cliente ou o servidor é desconectado da API.
- O API Gateway chama uma rota personalizada após a avaliação da expressão de seleção de rotas personalizada em relação à mensagem caso uma rota correspondente seja encontrada; a correspondência determina qual integração é invocada.
- O API Gateway chamará a rota `$default` se a expressão de seleção de rotas não puder ser avaliada em relação à mensagem ou se nenhuma rota correspondente for encontrada.

Para obter mais informações sobre as rotas `$connect` e `$disconnect`, consulte [the section called “Gerenciar usuários conectados e aplicações clientes”](#).

Para obter mais informações sobre a rota `$default` e rotas personalizadas, consulte [the section called “Invocar a integração do backend”](#).

## Enviar dados a aplicações de cliente conectadas

Os serviços de backend podem enviar dados para aplicativos conectados do cliente. É possível enviar dados fazendo o seguinte:

- Use uma integração para enviar uma resposta, que é exibida ao cliente por uma resposta de rota que você tiver definido.
- Você pode usar a API `@connections` para enviar uma solicitação POST: Para obter mais informações, consulte [the section called “Usar os comandos @connections em seu serviço de backend”](#).

## Códigos de status API de WebSocket

As APIs de WebSocket do Gateway da API usam os seguintes códigos de status para comunicação do servidor com o cliente, conforme descrito no [Registro de número de código de fechamento WebSocket](#):

1001

O API do Gateway retorna esse código de status quando o cliente fica inativo por 10 minutos ou atinge a vida útil máxima de conexão de 2 horas.

1003

O Gateway da API retorna esse código de status quando um endpoint recebe um tipo de mídia binária. Tipos de mídia binária não têm suporte para APIs de WebSocket.

1005

O Gateway da API retorna esse código de status se o cliente enviar um quadro de fechamento sem um código de encerramento.

1006

O Gateway da API retorna esse código de status se houver um fechamento inesperado da conexão, como a conexão TCP fechada sem um quadro de fechamento do WebSocket.

1008

O Gateway da API retorna esse código de status quando um endpoint recebe muitas solicitações de um cliente específico.

## 1009

O Gateway da API retorna esse código de status quando um endpoint recebe uma mensagem grande demais para ser processada.

## 1011

O Gateway da API retorna esse código de status quando há um erro interno do servidor.

## 1012

O Gateway da API retorna esse código de status se o serviço for reiniciado.

## Gerenciar usuários conectados e aplicações clientes: rotas **\$connect** e **\$disconnect**

A seção a seguir descreve como usar as rotas `$connect` e `$disconnect` para sua API de WebSocket.

### Tópicos

- [A rota `\$connect`](#)
- [Transmitir informações de conexão da rota `\$connect`](#)
- [A rota `\$disconnect`](#)

### A rota **\$connect**

Os aplicativos do cliente se conectam à sua API WebSocket ao enviar uma solicitação de atualização do WebSocket. Se a solicitação for bem-sucedida, a rota `$connect` é executada enquanto a conexão estiver sendo criada.

Como a conexão do WebSocket é uma conexão stateful, você pode configurar a autorização somente na rota `$connect`. AuthN/AuthZ será realizada somente pelo tempo de conexão.

Enquanto a execução de integração associada à rota `$connect` é concluída, a solicitação de atualização está pendente e a conexão real não será estabelecida. Se a solicitação `$connect` falhar (por exemplo, devido a uma falha AuthN/AuthZ ou falha de integração), a conexão não será estabelecida.

**Note**

Se a autorização falhar em `$connect`, a conexão não será estabelecida, e o cliente receberá uma resposta 401 ou 403.

A configuração de uma integração para `$connect` é opcional. Você deve considerar configurar uma integração `$connect` se:

- Deseja permitir que os clientes especifiquem subprotocolos usando o campo `Sec-WebSocket-Protocol`. Para ver um código demonstrativo, consulte [Configurar uma rota `\$connect` que requer um subprotocolo de WebSocket](#).
- Deseja receber notificação quando os clientes se conectarem.
- Deseja limitar as conexões ou controlar quem se conecta.
- Deseja que o backend envie mensagens de volta aos clientes usando uma URL de retorno de chamada.
- Deseja armazenar cada ID de conexão e outras informações em um banco de dados (por exemplo, Amazon DynamoDB).

## Transmitir informações de conexão da rota `$connect`

É possível usar integrações de proxy e não proxy para transmitir informações da rota `$connect` para um banco de dados ou outro AWS service (Serviço da AWS).

Como transmitir informações de conexão usando uma integração de proxy

É possível acessar as informações de conexão por uma integração de proxy do Lambda no evento. Use outro AWS service (Serviço da AWS) ou função do AWS Lambda para publicar na conexão.

A função do Lambda a seguir mostra como usar o objeto `requestContext` para registrar o ID da conexão, o nome do domínio, o nome do estágio e as strings 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}
```

Como transmitir informações de conexão usando uma integração que não seja de proxy

- É possível acessar as informações de conexão com uma integração não proxy. Configure a solicitação de integração e forneça um modelo de solicitação da API de WebSocket. O modelo de mapeamento [Velocity Template Language \(VTL\)](#) a seguir fornece uma solicitação de integração. Essa solicitação envia os seguintes detalhes para uma integração sem proxy:
  - ID da conexão
  - Nome de domínio
  - Nome da etapa
  - Caminho
  - Cabeçalhos

- Strings de consulta

Essa solicitação envia o ID da conexão, o nome do domínio, o nome do estágio, os caminhos, os cabeçalhos e as strings de consulta para uma integração sem proxy.

```
{
 "connectionId": "$context.connectionId",
 "domain": "$context.domainName",
 "stage": "$context.stage",
 "params": "$input.params()"
}
```

Para obter mais informações sobre a configuração de transformações de dados, consulte [the section called “Transformações de dados”](#).

Para concluir a solicitação de integração, defina `StatusCode: 200` para a resposta de integração. Para saber mais sobre como configurar uma resposta de integração, consulte [Configurar uma resposta de integração usando o console do API Gateway](#).

## A rota **\$disconnect**

A rota `$disconnect` é executada depois de a conexão ser encerrada.

A conexão pode ser encerrada pelo servidor ou pelo cliente. Como a conexão já está fechada quando é executado, `$disconnect` é um evento de melhor esforço. O API Gateway tentará o seu melhor para entregar o evento `$disconnect` à sua integração, mas não pode garantir a entrega.

O backend pode iniciar a desconexão ao utilizar a API `@connections`. Para obter mais informações, consulte [the section called “Usar os comandos @connections em seu serviço de backend”](#).

## Invocar sua integração de backend com o **\$default** Route e rotas personalizadas no API Gateway

A seção a seguir descreve como invocar sua integração de backend usando a rota `$default` ou uma rota personalizada para uma API de WebSocket.

### Tópicos

- [Usar rotas para processar mensagens](#)
- [A rota \\$default](#)
- [Rotas personalizadas](#)
- [Usar integrações da API WebSocket do API Gateway para se conectar à sua lógica de negócios](#)
- [Diferenças importantes entre APIs WebSocket e APIs REST](#)

## Usar rotas para processar mensagens

Nas APIs WebSocket do API Gateway, as mensagens podem ser enviadas do cliente para o serviço de backend e vice-versa. Ao contrário do modelo de solicitação/resposta HTTP, no WebSocket é possível que o backend envie mensagens para o cliente sem que o cliente realize qualquer ação.

As mensagens podem estar no formato JSON ou não. No entanto, somente as mensagens JSON podem ser roteadas para integrações específicas com base no conteúdo da mensagem. As mensagens que não estão no formato JSON não são transmitidas ao backend pela rota `$default`.

### Note

O API Gateway é compatível com cargas de mensagem de até 128 KB com quadros no tamanho máximo de 32 KB. Se uma mensagem exceder 32 KB, é necessário dividi-la em vários quadros, cada qual contendo 32 KB ou menos. Se uma mensagem (ou quadro) maior for recebida, a conexão será encerrada com o código 1009.

Atualmente, cargas binárias não são compatíveis. Se um quadro binário for recebido, a conexão será encerrada com o código 1003. No entanto, é possível converter cargas binárias para texto. Consulte [the section called “Tipos de mídia binária”](#).

Com APIs WebSocket no API Gateway, as mensagens JSON podem ser roteadas para executar um serviço de backend determinado com base no conteúdo da mensagem. Quando um cliente envia uma mensagem sobre sua conexão WebSocket, isso resulta em uma solicitação de rota para a API WebSocket. A solicitação será vinculada à rota com a chave da rota correspondente no API Gateway. É possível configurar uma solicitação de rota para uma API WebSocket no console do API Gateway, usando a AWS CLI ou um AWS SDK.

 Note

Na AWS CLI e nos AWS SDKs, você pode criar rotas antes ou depois de criar integrações. Atualmente, o console não oferece suporte à reutilização de integrações, portanto, você deve criar a rota primeiro e, em seguida, gerar a integração para essa rota.

É possível configurar o API Gateway para realizar a validação em uma solicitação de rota antes de prosseguir com a solicitação de integração. Se ocorrer uma falha na validação, o API Gateway não atenderá à solicitação sem chamar o backend, enviará uma resposta do gateway "Bad request body" semelhante à seguinte para o cliente e publicará os resultados da validação no CloudWatch Logs:

```
{"message" : "Bad request body", "connectionId": "{connectionId}", "messageId": "{messageId}"}
```

Isso reduz as chamadas desnecessárias para o backend e permite que você se concentre em outros requisitos da sua API.

Você também pode definir uma resposta de rota para suas rotas da API, permitindo comunicações bidirecionais. Uma resposta de rota descreve quais dados serão enviados ao cliente após a conclusão da integração de uma rota determinada. Não é necessário definir uma resposta para uma rota se, por exemplo, você deseja que um cliente envie mensagens para o backend sem receber uma resposta (comunicação unidirecional). No entanto, se você não fornecer uma resposta de rota, o API Gateway não enviará nenhuma informação sobre o resultado da sua integração aos clientes.

## A rota **\$default**

Cada API WebSocket do API Gateway pode ter uma rota **\$default**. Esse é um valor de roteamento especial que pode ser usado das seguintes formas:

- Você pode usá-lo em conjunto com chaves de roteamento definidas, para especificar uma rota de "fallback" (por exemplo, uma integração simulada genérica que retorna determinada mensagem de erro) para mensagens recebidas que não corresponderem a nenhuma das chaves de roteamento definidas.
- Você pode usá-lo sem qualquer chave de roteamento definida para especificar um modelo de proxy que delega o roteamento a um componente de backend.
- Você pode usá-lo para especificar uma rota para cargas que não estão no formato JSON.

## Rotas personalizadas

Se deseja invocar uma integração específica com base no conteúdo da mensagem, você pode fazê-lo ao criar uma rota personalizada.

A rota personalizada utiliza uma chave de roteamento e a integração que você especificar. Quando uma mensagem de entrada contém uma propriedade JSON, e essa propriedade é avaliada como um valor que corresponde ao valor da chave de roteamento, o API Gateway invoca a integração. (Para obter mais informações, consulte [the section called “Visão geral das APIs de WebSocket”](#).)

Por exemplo, suponha que você queira criar um aplicativo de salas de bate-papo. Você pode começar com a criação de uma API WebSocket cuja expressão de seleção de rotas é `$request.body.action`. Em seguida, você pode definir duas rotas: `joinroom` e `sendmessage`. Um aplicativo do cliente pode invocar a rota `joinroom` ao enviar uma mensagem como a seguinte:

```
{"action":"joinroom","roomname":"developers"}
```

Também pode invocar a rota `sendmessage` ao enviar uma mensagem como a seguinte:

```
{"action":"sendmessage","message":"Hello everyone"}
```

## Usar integrações da API WebSocket do API Gateway para se conectar à sua lógica de negócios

Após a configuração de uma rota para uma API WebSocket do API Gateway, é necessário especificar a integração a ser utilizada. Assim como ocorre com uma rota, que pode apresentar uma solicitação de rota e uma resposta de rota, uma integração pode apresentar uma solicitação de integração e uma resposta de integração. Uma solicitação de integração contém as informações esperadas pelo backend para processar a solicitação recebida de seu cliente. Uma resposta de integração contém os dados que o backend retorna para o API Gateway e que podem ser usados para elaborar uma mensagem a ser enviada ao cliente (se uma resposta de rota for definida).

Para obter mais informações sobre a configuração de integrações, consulte [the section called “Integrações”](#).

## Diferenças importantes entre APIs WebSocket e APIs REST

Integrações para APIs do WebSocket são semelhantes às integrações para APIs REST, exceto com relação às seguintes diferenças:

- Atualmente, no console do API Gateway, é necessário criar uma rota primeiro e, depois, criar uma integração como o destino da rota. No entanto, na API e CLI, você pode criar rotas e integrações de maneira independente, em qualquer ordem.
- Você pode usar uma única integração para várias rotas. Por exemplo, se você tiver um conjunto de ações que tenham uma relação estreita entre si, é recomendável que todas essas rotas sejam direcionadas a uma única função do Lambda. Em vez de definir os detalhes da integração várias vezes, você pode especificá-los uma vez e atribuí-los a cada uma das rotas relacionadas.

 Note

Atualmente, o console não oferece suporte à reutilização de integrações, portanto, você deve criar a rota primeiro e, em seguida, gerar a integração para essa rota. Na AWS CLI e nos AWS SDKs, você pode reutilizar uma integração definindo o destino da rota como um valor de "integrations/{*integration-id*}", em que *{integration-id}* é o ID exclusivo da integração a ser associada à rota.

- O API Gateway fornece várias [expressões de seleção](#) que podem ser usadas em suas rotas e integrações. Você não precisa depender do tipo de conteúdo para selecionar um modelo de entrada ou mapeamento de saída. Assim como ocorre com expressões de seleção de rotas, você pode definir uma expressão de seleção a ser avaliada pelo API Gateway para escolher o item correto. Todas elas serão recuadas para o modelo `$default` se um modelo correspondente não for encontrado.
- Nas solicitações de integração, a expressão de seleção do modelo é compatível com `$request.body.<json_path_expression>` e valores estáticos.
- Nas respostas de integração, a expressão de seleção do modelo é compatível com `$request.body.<json_path_expression>`, `$integration.response.statuscode`, `$integration.response.header.<headerName>` e valores estáticos.

No protocolo HTTP, em que solicitações e respostas são enviadas de forma síncrona, a comunicação é essencialmente unidirecional. No protocolo WebSocket, a comunicação é bidirecional. As respostas são assíncronas e não são necessariamente recebidas pelo cliente na mesma ordem em que as mensagens do cliente foram enviadas. Além disso, o backend pode enviar mensagens ao cliente.

 Note

Para uma rota que é configurada para utilizar a integração `AWS_PROXY` ou `LAMBDA_PROXY`, a comunicação é unidirecional, e o API Gateway não transmitirá automaticamente a resposta de backend para a resposta de rota. Por exemplo, no caso da integração `LAMBDA_PROXY`, o corpo retornado pela função do Lambda não será retornado ao cliente. Se deseja que o cliente receba respostas de integração, você deve definir uma resposta de rota para possibilitar a comunicação bidirecional.

## Expressões de seleção do WebSocket

O API Gateway usa expressões de seleção como uma maneira de avaliar o contexto de solicitação e resposta e produzir uma chave. A chave é utilizada para selecionar a partir de um conjunto de valores possíveis, normalmente fornecidos por você, que é o desenvolvedor da API. O conjunto exato de variáveis compatíveis varia de acordo com a expressão específica. Cada expressão é discutida a seguir com mais detalhes.

Para todas as expressões, o idioma segue o mesmo conjunto de regras:

- Uma variável é prefixada com "\$".
- As chaves podem ser usadas para definir explicitamente os limites da variável, por exemplo, "\${request.body.version}-beta".
- Várias variáveis são compatíveis, mas a avaliação ocorre somente uma vez (sem avaliação recursiva).
- Um cifrão (\$) pode ser recuado com "\". Isso é mais útil ao definir uma expressão que mapeia para a chave reservada \$default, por exemplo, "\"\$default".
- Em alguns casos, um formato padrão é obrigatório. Nesse caso, a expressão deve ser encapsulada com barras ("/"), por exemplo, "/2\d\d/" para corresponder aos códigos de status 2XX.

### Tópicos

- [Expressões de seleção de resposta de rotas](#)
- [Expressões de seleção de chave da API](#)
- [Expressões de seleção de mapeamento da API](#)
- [Resumo da expressão de seleção do WebSocket](#)

## Expressões de seleção de resposta de rotas

Uma [resposta de rota](#) é utilizada para modelar uma resposta do backend para o cliente. Para APIs WebSocket, uma resposta de rota é opcional. Quando definida, emite sinais para o API Gateway de que deve retornar uma resposta a um cliente após o recebimento de uma mensagem WebSocket.

A avaliação da expressão de seleção de resposta de rotas produz uma chave de resposta de rotas. Por fim, essa chave será usada para escolher um dos [RouteResponses](#) associados à API. No entanto, a única chave atualmente compatível é `$default`.

## Expressões de seleção de chave da API

Esta expressão é avaliada quando o serviço determina que uma dada solicitação deve continuar somente se o cliente fornecer uma [chave de API](#) válida.

Atualmente, os únicos dois valores compatíveis são `$request.header.x-api-key` e `$context.authorizer.usageIdentifierKey`.

## Expressões de seleção de mapeamento da API

Esta expressão é avaliada para determinar qual estágio de API é selecionado quando uma solicitação é feita por meio de um domínio personalizado.

Atualmente, o único valor compatível é `$request.basepath`.

## Resumo da expressão de seleção do WebSocket

A tabela a seguir resume os casos de uso para expressões de seleção em APIs WebSocket:

| Expressão de seleção                      | Avalia chave para           | Observações                                       | Exemplo de caso de uso                 |
|-------------------------------------------|-----------------------------|---------------------------------------------------|----------------------------------------|
| <code>Api.RouteSelectionExpression</code> | <code>Route.RouteKey</code> | <code>\$default</code> é compatível como uma rota | Roteie mensagens WebSocket com base no |



| Expressão de seleção                        | Avalia chave para                           | Observações                                                                                                                                              | Exemplo de caso de uso                                                 |
|---------------------------------------------|---------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------|
|                                             |                                             | genérica                                                                                                                                                 | contexto de uma solicitação do cliente.                                |
| <code>Route.ModelSelectionExpression</code> | Chave para <code>Route.RequestModels</code> | <p>Opcional</p> <p>Se fornecida para integração não proxy, a validação do modelo ocorre.</p> <p><code>\$default</code> é compatível como um genérico</p> | <p>Execute a validação de solicitação dinamicamente na mesma rota.</p> |

| Expressão de seleção                                 | Avalia chave para                                    | Observações                                                                                                                                                                                                                                       | Exemplo de caso de uso                                                                 |
|------------------------------------------------------|------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------|
| <code>Integration.TemplateSelectionExpression</code> | Chave para <code>Integration.RequestTemplates</code> | <p>Opcional. Pode ser fornecida para integração proxy visando manipular as cargas de entrada.</p> <p><code>\${request.body.jsonPath}</code> e valores estáticos são compatíveis.</p> <p><code>\$default</code> é compatível como um genérico.</p> | Manipular a solicitação do autor da chamada nas propriedades dinâmicas da solicitação. |

| Expressão de seleção                                 | Avalia chave para                             | Observações                                                                                                                                                                                                                     | Exemplo de caso de uso                                                                                                                                                          |
|------------------------------------------------------|-----------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>IntegrationResponse.SelectionExpression</code> | <code>IntegrationResponse.SelectionKey</code> | <p>Opcional. Pode ser fornecida para integração ou não proxy. Funciona como uma correspondência de padrão para mensagens de erro (do Lambda) ou códigos de status (de integração) es HTTP).</p> <p><code>\$default</code> é</p> | <p>Manipular a resposta do backend. Escolha a ação que deve ocorrer com base na resposta dinâmica do backend (por exemplo, manipular determinados erros de forma distinta).</p> |

| Expressão de seleção | Avalia chave para | Observações                                                                                       | Exemplo de caso de uso |
|----------------------|-------------------|---------------------------------------------------------------------------------------------------|------------------------|
|                      |                   | necessário para integrações não proxy visando atuar como um genérico para resposta bem-sucedidas. |                        |

| Expressão de seleção                                         | Avalia chave para                                             | Observações                                                                                                                                                                                                                     | Exemplo de caso de uso                                                                                           |
|--------------------------------------------------------------|---------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------|
| <code>IntegrationResponse.TemplateSelectionExpression</code> | Chave para <code>IntegrationResponse.ResponseTemplates</code> | <p>Opcional. Pode ser fornecida para integração proxy. O padrão \$ é compatível.</p> <p>Em alguns casos, uma propriedade de dinâmica da resposta pode ditar transformações diferentes na mesma rota e integração associada.</p> | <pre> \${request .body.js onPath} , \${integra tion.res onse.stat uscode} , \${integra tion.res onse.head </pre> |

| Expressão de seleção | Avalia chave para | Observações | Exemplo de caso de uso                                                                                                                                                                                                  |
|----------------------|-------------------|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                      |                   |             | <code>er.headerName} ,<br/>\${integration.response.multiHeaderValue.headerName} ,<br/>e<br/>valores<br/>estáticos<br/>são<br/>compatíveis.<br/><br/>\$default<br/>é<br/>compatível<br/>como<br/>um<br/>genérico.</code> |

| Expressão de seleção                                | Avalia chave para                                   | Observações                                                                                                                                                              | Exemplo de caso de uso |
|-----------------------------------------------------|-----------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|
| <code>Route.RouteResponseSelectionExpression</code> | <code>RouteResponse.RouteResponseKey</code>         | <p>Deve ser fornecido para iniciar a comunicação bidirecional para uma rota do WebSocket.</p> <p>Atualmente, este valor é restrito apenas ao <code>\$default</code>.</p> |                        |
| <code>RouteResponse.ModelSelectionExpression</code> | Chave para <code>RouteResponse.RequestModels</code> | Atualmente é incompatível.                                                                                                                                               |                        |

# Desenvolver APIs de WebSocket no API Gateway

Esta seção fornece detalhes sobre os recursos necessários do API Gateway durante o desenvolvimento de suas APIs do API Gateway.

Ao desenvolver a API do API Gateway, você decidirá uma série de características da API. Essas características dependem do caso de uso da sua API. Por exemplo, talvez você queira permitir que somente determinados clientes chamem sua API ou talvez queira disponibilizá-las a todos. É recomendável que uma chamada de API execute uma função do Lambda, faça uma consulta de banco de dados ou chame uma aplicação.

## Tópicos

- [Criar APIs de WebSocket no API Gateway](#)
- [Tipos de endereço IP para APIs de WebSocket no API Gateway](#)
- [Criar rotas para APIs de WebSocket no API Gateway](#)
- [Controlar e gerenciar o acesso a APIs de WebSocket no API Gateway](#)
- [Integrações para APIs de WebSocket no API Gateway](#)
- [Solicitar validação para APIs de WebSocket no API Gateway](#)
- [Transformações de dados para APIs de WebSocket no API Gateway](#)
- [Tipos de mídia binários para APIs de WebSocket no API Gateway](#)
- [Invocar APIs de WebSocket](#)

## Criar APIs de WebSocket no API Gateway

Você pode criar uma API de WebSocket no console do API Gateway usando o comando [create-api](#) da AWS CLI ou usando o comando `CreateApi` em um AWS SDK. Os procedimentos a seguir mostram como criar uma nova API WebSocket.

### Note

APIs de WebSocket são compatíveis somente com TLS 1.2 e TLS 1.3. Versões anteriores do TLS não são compatíveis.



## Criar uma API WebSocket usando comandos da AWS CLI

O comando [create-api](#) indicado abaixo cria uma API com a expressão de seleção de rota `$request.body.action`:

```
aws apigatewayv2 --region us-east-1 create-api --name "myWebSocketApi3" --protocol-type WEBSOCKET --route-selection-expression '$request.body.action'
```

A saída será exibida como a seguir:

```
{
 "ApiKeySelectionExpression": "$request.header.x-api-key",
 "Name": "myWebSocketApi3",
 "CreateDate": "2018-11-15T06:23:51Z",
 "ProtocolType": "WEBSOCKET",
 "RouteSelectionExpression": "'$request.body.action'",
 "ApiId": "aabbccddee"
}
```

## Criar uma API WebSocket usando o console do API Gateway

Você pode criar uma API WebSocket no console ao selecionar o protocolo WebSocket e atribuir um nome à API.

### Important

Depois de criar a API, você não pode alterar o protocolo escolhido. Não é possível converter uma API WebSocket em uma API REST ou vice-versa.

### Como criar uma API WebSocket usando o console do API Gateway

1. Inicie uma sessão no console do API Gateway e escolha Create API (Criar API).
2. Em WebSocket API (API WebSocket), escolha Build (Criar). Somente endpoints regionais são aceitos.
3. Em Nome da API, insira o nome da API.
4. Em Expressão de seleção de rotas, insira um valor. Por exemplo, `$request.body.action`.

Para obter mais informações sobre expressões de seleção de rota, consulte [the section called “”](#).

## 5. Execute um destes procedimentos:

- Selecione Criar API em branco para criar uma API sem rotas.
- Selecione Próximo para anexar rotas à API.

É possível anexar rotas depois de criar a API.

## Tipos de endereço IP para APIs de WebSocket no API Gateway

Ao criar uma API, você especifica o tipo de endereços IP que podem invocar sua API. É possível escolher IPv4 para resolver endereços IPv4 e invocar sua API ou escolher pilha dupla para permitir que endereços IPv4 e IPv6 invoquem sua API. Recomendamos que você defina o tipo de endereço IP como pilha dupla para aliviar o esgotamento do espaço IP ou para seu procedimento de segurança. Para ter mais informações sobre os benefícios de um tipo de endereço IP de pilha dupla, consulte [IPv6 na AWS](#).

### Considerações sobre tipos de endereço IP

As seguintes considerações podem afetar o uso de tipos de endereço IP:

- O tipo de endereço IP padrão para todas as APIs de WebSocket é IPv4.
- Se você alterar o tipo de endereço IP de uma API existente de IPv4 para pilha dupla, confirme se todas as políticas que controlam o acesso às suas APIs foram atualizadas para considerar as chamadas IPv6. Quando você altera o tipo de endereço IP, a alteração entra em vigor imediatamente.
- Sua API pode ser associada a um nome de domínio personalizado com um tipo de endereço IP diferente do da sua API. Se você desabilitar seu endpoint de API padrão, isso poderá afetar como os chamadores podem invocar sua API.

### Alterar o tipo de endereço IP de uma API de WebSocket

É possível alterar o tipo de endereço IP atualizando a configuração da API. É possível atualizar a configuração da API usando o Console de gerenciamento da AWS, a AWS CLI, o CloudFormation ou um SDK da AWS. Se você alterar o tipo de endereço IP da API, não poderá reimplantá-la para que as alterações entrem em vigor.

## Console de gerenciamento da AWS

Como alterar o tipo de endereço IP de uma API de WebSocket

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API de WebSocket.
3. Selecione Configurações da API e, depois, escolha Editar.
4. Em Tipo de endereço IP, escolha IPv4 ou Pilha dupla.
5. Escolha Salvar.

A alteração na configuração da sua API entrará em vigor imediatamente.

## AWS CLI

O comando [update-api](#) a seguir atualiza uma API para ter um tipo de endereço IP de pilha dupla:

```
aws apigatewayv2 update-api \
 --api-id abcd1234 \
 --ip-address-type dualstack
```

A saída será exibida da seguinte forma:

```
{
 "ApiEndpoint": "https://abcd1234.execute-api.us-east-1.amazonaws.com",
 "ApiId": "abcd1234",
 "ApiKeySelectionExpression": "$request.header.x-api-key",
 "CreateDate": "2025-02-04T22:20:20+00:00",
 "DisableExecuteApiEndpoint": false,
 "Name": "My-WebSocket-API",
 "ProtocolType": "WEBSOCKET",
 "RouteSelectionExpression": "$request.method $request.path",
 "Tags": {},
 "NotificationUri": [],
 "IpAddressType": "dualstack"
}
```

## Criar rotas para APIs de WebSocket no API Gateway

Na sua API WebSocket, mensagens JSON de entrada são direcionadas para integrações de backend com base nas rotas que você configurar. (Mensagens que não apresentam o formato JSON são direcionadas para a rota `$default` que você configurar).

Uma rota inclui uma chave de roteamento, que é o valor esperado quando uma expressão de seleção de rotas é avaliada. O `routeSelectionExpression` é um atributo definido em nível de API. Ele especifica uma propriedade JSON que deve estar presente na carga da mensagem. Para obter mais informações sobre expressões de seleção de rota, consulte [the section called “”](#).

Por exemplo, se as suas mensagens JSON contiverem uma propriedade `action` e você desejar executar ações diferentes de acordo com essa propriedade, sua expressão de seleção de rotas poderá ser `${request.body.action}`. A tabela de roteamento deve especificar qual ação executar ao corresponder o valor da propriedade `action` com os valores de chave de rotas personalizada que você definiu na tabela.

Há três rotas predefinidas que podem ser usadas: `$connect`, `$disconnect` e `$default`. Além disso, você pode criar rotas personalizadas.

- O API Gateway chama a rota `$connect` quando uma conexão persistente entre o cliente e uma API WebSocket está sendo iniciada.
- O API Gateway chama a rota `$disconnect` quando o cliente ou o servidor é desconectado da API.
- O API Gateway chama uma rota personalizada após a avaliação da expressão de seleção de rotas personalizada em relação à mensagem caso uma rota correspondente seja encontrada; a correspondência determina qual integração é invocada.
- O API Gateway chamará a rota `$default` se a expressão de seleção de rotas não puder ser avaliada em relação à mensagem ou se nenhuma rota correspondente for encontrada.

### Expressões de seleção de rota

Uma expressão de seleção de rota é avaliada quando o serviço está selecionando a rota para seguir para uma mensagem recebida. O serviço usa a rota cuja `routeKey` corresponde exatamente ao valor avaliado. Se não existir nenhuma correspondência e uma rota com a chave `$default` existir, essa rota será selecionada. Se nenhuma rota corresponder ao valor avaliado e não existir uma rota `$default`, o serviço retornará uma mensagem de erro. Para APIs com base em WebSocket, a expressão deve ser da forma `${request.body.{path_to_body_element}}`.

Por exemplo, suponha que você está enviando a seguinte mensagem JSON:

```
{
 "service" : "chat",
 "action" : "join",
 "data" : {
 "room" : "room1234"
 }
}
```

Você pode selecionar o comportamento da API com base na propriedade `action`. Nesse caso, você pode definir a seguinte expressão de seleção de rotas:

```
$request.body.action
```

Neste exemplo, `request.body` refere-se à carga JSON da sua mensagem, e `.action` é uma expressão [JSONPath](#). Você pode usar qualquer expressão de caminho JSON após `request.body`, mas lembre-se de que o resultado será transformado em string. Por exemplo, se a expressão JSONPath retorna uma matriz de dois elementos, que serão apresentadas como a string `"[item1, item2]"`. Por esse motivo, é uma boa prática ter sua expressão avaliada como um valor e não uma matriz ou um objeto.

Você pode simplesmente usar um valor estático, ou utilizar várias variáveis. A tabela a seguir mostra exemplos e seus resultados avaliados em relação à carga anterior.

| Expressão                             | Resultado avaliado | Descrição                  |
|---------------------------------------|--------------------|----------------------------|
| <code>\$request.body.action</code>    | join               | Uma variável desempacotada |
| <code>\${request.body.action}</code>  | join               | Uma variável empacotada    |
| <code>\${request.body.service}</code> | chat/join          | Várias variáveis           |

| Expressão                                                         | Resultado avaliado     | Descrição                                                            |
|-------------------------------------------------------------------|------------------------|----------------------------------------------------------------------|
| <code>vice}/{request.body.action}</code>                          |                        | com valores estáticos                                                |
| <code>\${request.body.action}-\${request.body.invalidPath}</code> | <code>join-</code>     | Se o JSONPath não for encontrado, a variável será resolvida como "". |
| <code>action</code>                                               | <code>action</code>    | Valor estático                                                       |
| <code>\\$default</code>                                           | <code>\$default</code> | Valor estático                                                       |

O resultado avaliado é usado para encontrar uma rota. Se houver uma rota com uma chave de rota correspondente, a rota será selecionada para processar a mensagem. Se nenhuma rota correspondente for encontrada, o API Gateway tentará encontrar a rota `$default`, se disponível. Se a rota `$default` não estiver definida, o API Gateway retornará um erro.

## Configurar rotas para uma API WebSocket no API Gateway

Ao criar uma nova API WebSocket pela primeira vez, há três rotas predefinidas: `$connect`, `$disconnect` e `$default`. É possível criá-las ao utilizar o console, a API ou a AWS CLI. Se desejar, você pode criar rotas personalizadas. Para obter mais informações, consulte [the section called “Visão geral das APIs de WebSocket”](#).

### Note

Na CLI, você pode criar rotas antes ou depois de gerar integrações, sendo possível reutilizar a mesma integração para várias rotas.

## Criar uma rota usando o console do API Gateway

### Como criar uma rota usando o console do API Gateway

1. Inicie uma sessão no console do API Gateway, escolha a API e selecione Routes (Rotas).
2. Selecione Criar rota.
3. Em Chave de rota, insira o nome da chave de rota. É possível criar as rotas predefinidas (\$connect, \$disconnect e \$default) ou uma rota personalizada.

#### Note

Ao criar uma rota personalizada, não use o prefixo \$ no nome da chave de rota. Este prefixo está reservado para rotas predefinidas.

4. Selecione e configure o tipo de integração para a rota. Para obter mais informações, consulte [the section called “Configurar uma solicitação de integração de API WebSocket usando o console do API Gateway”](#).

### Criar uma rota usando a AWS CLI

O comando [create-route](#) indicado abaixo cria uma rota:

```
aws apigatewayv2 --region us-east-1 create-route --api-id aabbccddee --route-key $default
```

A saída será exibida da seguinte forma:

```
{
 "ApiKeyRequired": false,
 "AuthorizationType": "NONE",
 "RouteKey": "$default",
 "RouteId": "1122334"
}
```

### Especificar as configurações de solicitação de rota para **\$connect**

Quando você configurar a rota \$connect para a sua API, as configurações opcionais a seguir serão disponibilizadas para habilitar a autorização para sua API. Para obter mais informações, consulte [the section called “A rota \\$connect”](#).

- **Autorização:** Se a autorização não for necessária, você pode especificar `NONE`. Caso contrário, especifique:
  - `AWS_IAM` para utilizar políticas padrão do AWS IAM para controlar o acesso à sua API.
  - `CUSTOM` para implementar a autorização para uma API especificando uma função de autorizador do Lambda criada anteriormente. O autorizador pode residir em sua própria conta da AWS ou em outra conta da AWS. Para obter mais informações sobre autorizadores do Lambda, consulte [Usar os autorizadores do API Gateway Lambda](#).

 **Note**

No console do API Gateway, a configuração `CUSTOM` é visível somente após a configuração de uma função do autorizador, conforme descrito em [the section called “Configurar um autorizador do Lambda \(console\)”](#).

 **Important**

A configuração de Autorização é aplicada à API completa, e não apenas à rota `$connect`. A rota `$connect` protege as outras rotas, visto que é chamada em cada conexão.

- **Chave de API obrigatória:** opcionalmente, você pode exigir uma chave de API para uma rota `$connect` da API. Você pode usar chaves da API com planos de uso para controlar e rastrear o acesso às APIs. Para obter mais informações, consulte [the section called “Planos de uso”](#).

Configurar a solicitação de rota **\$connect** usando o console do API Gateway

Para configurar uma solicitação de rota `$connect` para uma API WebSocket usando o console do API Gateway:

1. Inicie uma sessão no console do API Gateway, escolha a API e selecione Routes (Rotas).
2. Em Rotas, selecione `$connect` ou crie uma rota `$connect` seguindo [the section called “Criar uma rota usando o console do API Gateway”](#).
3. Na seção Configurações de solicitação de rota, selecione Editar.
4. Em Autorização, selecione um tipo de autorizador.
5. Para exigir uma API para a rota `$connect`, selecione Exigir chave de API.
6. Escolha Salvar alterações.



## Configurar respostas de rotas para APIs de WebSocket no API Gateway

As rotas do WebSocket podem ser configuradas para comunicação bidirecional ou unidirecional. O API Gateway não transmitirá a resposta de back-end para a resposta de rota, a menos que você configure uma resposta de rota.

### Note

Você só pode definir a resposta de rota `$default` para as APIs do WebSocket. É possível usar uma resposta de integração para manipular a resposta de um serviço de back-end. Para obter mais informações, consulte [the section called “Visão geral das respostas de integração”](#).

É possível configurar respostas de rota e expressões de seleção de resposta usando o console do API Gateway ou a AWS CLI ou um SDK da AWS.

Para obter mais informações sobre expressões de seleção de respostas de rota, consulte [the section called “”](#).

### Tópicos

- [Configurar uma resposta de rota usando o console do API Gateway](#)
- [Configurar uma resposta de rota usando a AWS CLI](#)

### Configurar uma resposta de rota usando o console do API Gateway

Depois de criar uma API de WebSocket e anexar uma função proxy do Lambda à rota padrão, você pode configurar a resposta da rota usando o console do API Gateway:

1. Faça login no console do API Gateway e selecione uma API do WebSocket com uma integração de função proxy do Lambda na rota `$default`.
2. Em Routes (Rotas), selecione a rota `$default`.
3. Selecione Habilitar comunicação bidirecional.
4. Escolha Implantar API.
5. Implante a API em um estágio.

Use o comando [wscat](#) a seguir para se conectar à sua API. Para obter mais informações sobre o `wscat`, consulte [the section called “Use wscat para se conectar a uma API do WebSocket e enviar mensagens a ela”](#).

```
wscat -c wss://api-id.execute-api.us-east-2.amazonaws.com/test
```

Pressione o botão enter para chamar a rota padrão. O corpo de sua função do Lambda deve retornar.

Configurar uma resposta de rota usando a AWS CLI

O comando [create-route-response](#) a seguir cria uma resposta de rota para a rota `$default`. Você pode identificar o ID da API e o ID da rota usando os comandos [get-apis](#) e [get-routes](#).

```
aws apigatewayv2 create-route-response \
 --api-id aabbccdde \
 --route-id 1122334 \
 --route-response-key '$default'
```

A saída será exibida da seguinte forma:

```
{
 "RouteResponseId": "abcdef",
 "RouteResponseKey": "$default"
}
```

Configurar uma rota **`$connect`** que requer um subprotocolo de WebSocket

Os clientes podem usar o campo `Sec-WebSocket-Protocol` para solicitar um [subprotocolo WebSocket](#) durante a conexão com sua API WebSocket. É possível configurar uma integração para a rota `$connect` para permitir conexões somente se um cliente solicitar um subprotocolo compatível com sua API.

A função demonstrativa do Lambda a seguir retorna o cabeçalho `Sec-WebSocket-Protocol` aos clientes. A função só estabelecerá uma conexão com sua API se o cliente especificar o subprotocolo `myprotocol`.

Para obter um modelo do CloudFormation que cria essa API de exemplo e integração de proxy do 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;
}
```

Você só poderá usar [wscat](#) para testar se sua API permite conexões se um cliente solicitar um subprotocolo compatível com sua API. Os comandos a seguir usam o sinalizador `-s` para especificar subprotocolos durante a conexão.

O comando a seguir tenta uma conexão com um subprotocolo incompatível. Como o cliente especificou o subprotocolo `chat1`, a integração do Lambda retorna um erro 400 e a conexão não obtém êxito.

```
wscat -c wss://api-id.execute-api.region.amazonaws.com/beta -s chat1
error: Unexpected server response: 400
```

O comando a seguir inclui um subprotocolo compatível na solicitação de conexão. A integração do Lambda permite a conexão.

```
wscat -c wss://api-id.execute-api.region.amazonaws.com/beta -s chat1,myprotocol
connected (press CTRL+C to quit)
```

Para saber mais sobre como invocar APIs WebSocket, consulte [Invocar APIs de WebSocket](#).

## Controlar e gerenciar o acesso a APIs de WebSocket no API Gateway

O API Gateway é compatível com vários mecanismos para controlar e gerenciar o acesso à sua API WebSocket.

Os mecanismos a seguir podem ser usados para autenticação e autorização:

- As funções e políticas padrão do AWS IAM oferecem controles de acesso flexíveis e robustos. É possível usar as funções e políticas do IAM para controlar quem pode criar e gerenciar suas APIs, bem como quem pode chamá-las. Para obter mais informações, consulte [Controlar o acesso a APIs de WebSocket com autorização do IAM](#).
- As tags do IAM podem ser usadas com as políticas do IAM para controlar o acesso. Para obter mais informações, consulte [Usar tags para controlar o acesso aos recursos da API REST do API Gateway](#).
- Autorizadores do Lambda são funções do Lambda que controlam o acesso a APIs. Para obter mais informações, consulte [Controlar o acesso a APIs de WebSocket com autorizadores REQUEST do AWS Lambda](#).

Para melhorar seu procedimento de segurança, recomendamos configurar um autorizador para a rota \$connect em todas as suas APIs de WebSocket. Você pode precisar fazer isso para cumprir vários requisitos de conformidade. Para ter mais informações, consulte [Amazon API Gateway controls](#) no Guia do usuário do AWS Security Hub.

### Tópicos

- [Controlar o acesso a APIs de WebSocket com autorização do IAM](#)
- [Controlar o acesso a APIs de WebSocket com autorizadores REQUEST do AWS Lambda](#)

## Controlar o acesso a APIs de WebSocket com autorização do IAM

A autorização do IAM em APIs WebSocket é semelhante à utilizada para [APIs REST](#), com as seguintes exceções:

- A ação `execute-api` oferece suporte a `ManageConnections`, além de ações existentes (`Invoke`, `InvalidateCache`). `ManageConnections` controla o acesso à API `@connections`.
- As rotas do WebSocket utilizam um formato diferente de ARN:

```
arn:aws:execute-api:region:account-id:api-id/stage-name/route-key
```

- A API `@connections` utiliza o mesmo formato de ARN presente nas APIs REST:

```
arn:aws:execute-api:region:account-id:api-id/stage-name/POST/@connections
```

### Important

Ao usar a [Autorização do IAM](#), é necessário assinar solicitações com o [Signature Version 4 \(SigV4\)](#).

Por exemplo, você pode configurar a política a seguir ao cliente: Este exemplo permite que todas as pessoas enviem uma mensagem (`Invoke`) a todas as rotas, exceto para uma rota secreta no estágio `prod`, além de impedir que qualquer pessoa envie uma mensagem aos clientes conectados (`ManageConnections`) para todos os estágios.

### 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/*"
]
 }
]
}

```

## Controlar o acesso a APIs de WebSocket com autorizadores REQUEST do AWS Lambda

Uma função do autorizador do Lambda em APIs WebSocket é semelhante à utilizada para [APIs REST](#), com as seguintes exceções:

- Você só pode usar uma função de autorizador do Lambda para a rota `$connect`.
- Você não pode utilizar variáveis de caminho (`event.pathParameters`), afinal, o caminho é fixo.
- `event.methodArn` é diferente de sua API REST equivalente, visto que não possui um método HTTP. No caso de `$connect`, `methodArn` termina com `"$connect"`:

```
arn:aws:execute-api:region:account-id:api-id/stage-name/$connect
```

- As variáveis de contexto em `event.requestContext` são diferentes das utilizadas para APIs REST.

O exemplo a seguir mostra uma entrada para um autorizador REQUEST de uma 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"
 }
}
```



A função demonstrativa do autorizador do Lambda a seguir é uma versão WebSocket da função do autorizador do Lambda para APIs REST em [the section called “Exemplos adicionais de funções do autorizador do Lambda”](#):

## Node.js

```
// A simple REQUEST authorizer example to demonstrate how to use request
// parameters to allow or deny a request. In this example, a request is
// authorized if the client-supplied HeaderAuth1 header and QueryString1 query
parameter
// in the request context match the specified values of
// of 'headerValue1' and 'queryValue1' respectively.
 export const handler = function(event, context, callback) {
 console.log('Received event:', JSON.stringify(event, null, 2));

// Retrieve request parameters from the Lambda function input:
var headers = event.headers;
var queryStringParameters = event.queryStringParameters;
var stageVariables = event.stageVariables;
var requestContext = event.requestContext;

// Parse the input for the parameter values
var tmp = event.methodArn.split(':');
var apiGatewayArnTmp = tmp[5].split('/');
var awsAccountId = tmp[4];
var region = tmp[3];
var ApiId = apiGatewayArnTmp[0];
var stage = apiGatewayArnTmp[1];
var route = apiGatewayArnTmp[2];

// Perform authorization to return the Allow policy for correct parameters and
// the 'Unauthorized' error, otherwise.
var authResponse = {};
var condition = {};
 condition.IpAddress = {};

if (headers.HeaderAuth1 === "headerValue1"
 && queryStringParameters.QueryString1 === "queryValue1") {
 callback(null, generateAllow('me', event.methodArn));
} else {
 callback(null, generateDeny('me', event.methodArn));
}
}
```

```
// Helper function to generate an IAM policy
var generatePolicy = function(principalId, effect, resource) {
 // Required output:
 var authResponse = {};
 authResponse.principalId = principalId;
 if (effect && resource) {
 var policyDocument = {};
 policyDocument.Version = '2012-10-17'; // default version
 policyDocument.Statement = [];
 var statementOne = {};
 statementOne.Action = 'execute-api:Invoke'; // default action
 statementOne.Effect = effect;
 statementOne.Resource = resource;
 policyDocument.Statement[0] = statementOne;
 authResponse.policyDocument = policyDocument;
 }
 // Optional output with custom properties of the String, Number or Boolean type.
 authResponse.context = {
 "stringKey": "stringval",
 "numberKey": 123,
 "booleanKey": true
 };
 return authResponse;
}

var generateAllow = function(principalId, resource) {
 return generatePolicy(principalId, 'Allow', resource);
}

var generateDeny = function(principalId, resource) {
 return generatePolicy(principalId, 'Deny', resource);
}
```

## Python

```
A simple REQUEST authorizer example to demonstrate how to use request
parameters to allow or deny a request. In this example, a request is
authorized if the client-supplied HeaderAuth1 header and QueryString1 query
parameter
in the request context match the specified values of
of 'headerValue1' and 'queryValue1' respectively.
```

```
import json

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

 # Retrieve request parameters from the Lambda function input:
 headers = event['headers']
 queryStringParameters = event['queryStringParameters']
 stageVariables = event['stageVariables']
 requestContext = event['requestContext']

 # Parse the input for the parameter values
 tmp = event['methodArn'].split(':')
 apiGatewayArnTmp = tmp[5].split('/')
 awsAccountId = tmp[4]
 region = tmp[3]
 ApiId = apiGatewayArnTmp[0]
 stage = apiGatewayArnTmp[1]
 route = apiGatewayArnTmp[2]

 # Perform authorization to return the Allow policy for correct parameters
 # and the 'Unauthorized' error, otherwise.

 authResponse = {}
 condition = {}
 condition['IpAddress'] = {}

 if (headers['HeaderAuth1'] ==
 "headerValue1" and queryStringParameters["QueryString1"] ==
"queryValue1"):
 response = generateAllow('me', event['methodArn'])
 print('authorized')
 return json.loads(response)
 else:
 response = generateDeny('me', event['methodArn'])
 print('unauthorized')
 return json.loads(response)

 # Help function to generate IAM policy

def generatePolicy(principalId, effect, resource):
 authResponse = {}
```

```
authResponse['principalId'] = principalId
if (effect and resource):
 policyDocument = {}
 policyDocument['Version'] = '2012-10-17'
 policyDocument['Statement'] = []
 statementOne = {}
 statementOne['Action'] = 'execute-api:Invoke'
 statementOne['Effect'] = effect
 statementOne['Resource'] = resource
 policyDocument['Statement'] = [statementOne]
 authResponse['policyDocument'] = policyDocument

authResponse['context'] = {
 "stringKey": "stringval",
 "numberKey": 123,
 "booleanKey": True
}

authResponse_JSON = json.dumps(authResponse)

return authResponse_JSON

def generateAllow(principalId, resource):
 return generatePolicy(principalId, 'Allow', resource)

def generateDeny(principalId, resource):
 return generatePolicy(principalId, 'Deny', resource)
```

Para configurar a função anterior do Lambda como uma função do autorizador REQUEST para uma API WebSocket, siga o mesmo procedimento para [APIs REST](#).

Para configurar a rota \$connect a fim de esse autorizador do Lambda no console, selecione ou crie a rota \$connect. Na seção Configurações de solicitação de rota, selecione Editar. Selecione o autorizador no menu suspenso Autorização e escolha Salvar alterações.

Para testar o autorizador, é necessário uma conexão. O autorizador da alteração no \$connect não afeta o cliente já conectado. Quando você se conectar à API WebSocket, será necessário fornecer valores para todas as origens de identidade configuradas. Por exemplo, você pode se conectar ao enviar uma string de consulta e cabeçalho válidos usando wscat como no exemplo a seguir:

```
wscat -c 'wss://myapi.execute-api.us-east-1.amazonaws.com/beta?
QueryString=queryValue1' -H HeaderAuth1:headerValue1
```

Ao tentar se conectar sem um valor de identidade válido, você receberá uma resposta 401:

```
wscat -c wss://myapi.execute-api.us-east-1.amazonaws.com/beta
error: Unexpected server response: 401
```

## Integrações para APIs de WebSocket no API Gateway

Após a configuração de uma rota de API, você deve integrá-la a um endpoint no backend. Um endpoint de backend também é conhecido como endpoint de integração e pode ser uma função do Lambda, um endpoint HTTP ou uma ação de serviço da AWS. A integração da API tem uma solicitação e uma resposta de integração.

Nesta seção, você pode aprender como configurar solicitações e respostas de integração para sua API WebSocket.

### Tópicos

- [Configurar uma solicitação de integração da API de WebSocket no API Gateway](#)
- [Configurar uma resposta de integração da API de WebSocket no API Gateway](#)

## Configurar uma solicitação de integração da API de WebSocket no API Gateway

A configuração de uma solicitação de integração envolve o seguinte:

- Escolher uma chave de roteamento a ser integrada ao backend.
- Especificar o endpoint de back-end a ser invocado. As APIs do WebSocket são compatíveis com os seguintes tipos de integração:
  - AWS\_PROXY
  - AWS
  - HTTP\_PROXY
  - HTTP
  - MOCK

Para obter mais informações sobre tipos de integração, consulte [IntegrationType](#) na API REST do API Gateway V2.

- Configurar como transformar os dados da solicitação de rota, se necessário, para os dados da solicitação de integração, especificando um ou mais modelos de solicitação.

Configurar uma solicitação de integração de API WebSocket usando o console do API Gateway

Como adicionar uma solicitação de integração para uma rota em uma API WebSocket usando o console do API Gateway

1. Inicie uma sessão no console do API Gateway, escolha a API e selecione Routes (Rotas).
2. Em Rotas, selecione a rota.
3. Escolha a guia Solicitação de integração e, na seção Configurações de solicitação de integração, selecione Editar.
4. Em Tipo de integração, selecione uma das seguintes opções:
  - Selecione Função do Lambda somente se a API for integrada a uma função do AWS Lambda que você já criou nesta ou em outra conta.

Para criar uma função do Lambda no AWS Lambda, definir uma permissão do recurso na função do Lambda ou executar qualquer outra ação de serviço do Lambda, escolha Serviço da AWS.

- Escolha HTTP se a sua API for integrada a um endpoint HTTP existente. Para obter mais informações, consulte [Integrações de HTTP para APIs REST no API Gateway](#).
  - Escolha Mock (Simulação) se deseja gerar respostas de API diretamente do API Gateway, sem a necessidade de um backend de integração. Para obter mais informações, consulte [Simular integrações para APIs REST no API Gateway](#).
  - Selecione Serviço da AWS se a API for integrada a um serviço da AWS.
  - Selecione Link de VPC se a API utilizar VpcLink como um endpoint de integração privada. Para obter mais informações, consulte [Configurar uma integração privada](#).
5. Se você escolheu Função do Lambda, faça o seguinte:
    - a. Em Usar a integração de proxy do Lambda, marque a caixa de seleção se você pretende usar a [Integração de proxy do Lambda](#) ou a [Integração de proxy do Lambda entre contas](#).
    - b. Em Função do Lambda, especifique a função de uma das seguintes formas:
      - Se a função do Lambda estiver na mesma conta, insira o nome da função e selecione-a na lista suspensa.

 Note

O nome da função pode, opcionalmente, incluir seu alias ou sua especificação de versão, como em HelloWorld, HelloWorld:1 ou HelloWorld:alpha.

- Se a função estiver em uma conta diferente, insira o ARN da função.
- c. Para usar o valor de tempo limite padrão de 29 segundos, deixe a opção Tempo limite padrão ativada. Para definir um tempo limite personalizado, selecione Tempo limite padrão e insira um valor de tempo limite entre 50 e 29000 milissegundos.
- 6. Se você escolher HTTP, siga as instruções na etapa 4 de [the section called “Configurar uma solicitação de integração usando o console”](#).
- 7. Se você escolheu Simulação, prossiga para a etapa de Modelos de solicitação.
- 8. Se você escolheu Serviço da AWS, siga as instruções na etapa 6 de [the section called “Configurar uma solicitação de integração usando o console”](#).
- 9. Se você escolheu Link de VPC, faça o seguinte:
  - a. Em Usar a integração de proxy, marque a caixa de seleção se desejar que as solicitações sejam encaminhadas por proxy ao endpoint do VPCLink.
  - b. Em HTTP method (Método HTTP), escolha o tipo de método HTTP mais parecido com o método no backend HTTP.
  - c. Na lista suspensa Link de VPC, selecione um link de VPC. Você pode selecionar [Use Stage Variables] e inserir `${stageVariables.vpcLinkId}` na caixa de texto abaixo da lista.

É possível definir a variável do estágio vpcLinkId depois de implantar a API em um estágio e definir o valor como o ID do VpcLink.
  - d. Em Endpoint URL (URL de endpoint), insira o URL do backend HTTP que você deseja que essa integração use.
  - e. Para usar o valor de tempo limite padrão de 29 segundos, deixe a opção Tempo limite padrão ativada. Para definir um tempo limite personalizado, selecione Tempo limite padrão e insira um valor de tempo limite entre 50 e 29000 milissegundos.
- 10. Escolha Salvar alterações.
- 11. Em Modelos de solicitação, faça o seguinte:
  - a. Em Expressão de seleção de modelos, em Modelos de solicitação, selecione Editar.

- b. Insira uma Expressão de seleção de modelos. Use uma expressão que o API Gateway procura na carga útil de mensagens. Ela será avaliada se for encontrada, e o resultado é um valor de chave de modelo que é utilizado para selecionar o modelo de mapeamento de dados a ser aplicado aos dados na carga da mensagem. Você criará o modelo de mapeamento de dados na próxima etapa. Selecione Editar para salvar as alterações.
- c. Selecione Criar modelo para criar o modelo de mapeamento de dados. Em Chave de modelo, insira o valor de chave de modelo que é utilizado para selecionar o modelo de mapeamento de dados a ser aplicado aos dados na carga útil de mensagens. Depois, insira um modelo de mapeamento. Selecione Criar modelo.

Para obter mais informações sobre expressões de seleção de modelo, consulte [the section called “Expressões de seleção de modelo”](#).

## Configurar uma solicitação de integração usando a AWS CLI

Você pode configurar uma solicitação de integração para uma rota em uma API WebSocket usando a AWS CLI como no exemplo a seguir, que cria uma integração simulada:

1. Crie um arquivo denominado `integration-params.json` com o seguinte conteúdo:

```
{"PassthroughBehavior": "WHEN_NO_MATCH", "TimeoutInMillis": 29000,
 "ConnectionType": "INTERNET", "RequestTemplates": {"application/json":
 "{\"statusCode\":200}"}, "IntegrationType": "MOCK"}
```

2. Use o comando [create-integration](#) indicado abaixo para criar a integração simulada.

```
aws apigatewayv2 --region us-east-1 create-integration --api-id aabbccdde --cli-
input-json file://integration-params.json
```

A saída será exibida da seguinte forma:

```
{
 "PassthroughBehavior": "WHEN_NO_MATCH",
 "TimeoutInMillis": 29000,
 "ConnectionType": "INTERNET",
 "IntegrationResponseSelectionExpression": "${response.statuscode}",
 "RequestTemplates": {
 "application/json": "{\"statusCode\":200}"
 },
}
```



```

 "IntegrationId": "0abcdef",
 "IntegrationType": "MOCK"
 }

```

Também é possível configurar uma solicitação de integração para uma integração de proxy usando a AWS CLI.

1. Crie uma função do Lambda no console do Lambda e forneça uma função de execução básica do Lambda.
2. Use o comando [create-integration](#) indicado abaixo para criar a integração.

```

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

```

A saída será exibida da seguinte forma:

```

{
 "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 uma função do Lambda para integração de proxy de APIs de WebSocket

Com a integração de proxy do Lambda, o API Gateway mapeia toda a solicitação do cliente para o parâmetro event de entrada da função do Lambda de back-end. O seguinte exemplo mostra a estrutura do evento de entrada da rota \$connect e o evento de entrada da rota \$disconnect que o API Gateway envia para uma integração de proxy do 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
}
```

## Configurar uma resposta de integração da API de WebSocket no API Gateway

A seção a seguir oferece uma breve visão geral das respostas de integração da API de WebSocket e como configurar uma resposta de integração de uma API de WebSocket.

### Tópicos

- [Visão geral das respostas de integração](#)
- [Respostas de integração para comunicação bidirecional](#)
- [Configurar uma resposta de integração usando o console do API Gateway](#)

- [Configurar uma resposta de integração usando a AWS CLI](#)

## Visão geral das respostas de integração

A resposta de integração do API Gateway é uma maneira de modelar e manipular a resposta de um serviço de backend. Existem algumas diferenças na configuração de uma resposta de integração entre uma API REST e uma API WebSocket, mas, conceitualmente, o comportamento é o mesmo.

As rotas do WebSocket podem ser configuradas para comunicação bidirecional ou unidirecional.

- Quando uma rota está configurada para comunicação bidirecional, uma resposta de integração permite que você configure as transformações na carga útil da mensagem retornada, semelhante às respostas de integração para APIs REST.
- Se uma rota está configurada para comunicação unidirecional, independentemente de qualquer configuração da resposta de integração, nenhuma resposta será retornada pelo canal do WebSocket após o processamento da mensagem.

O API Gateway não transmitirá a resposta de back-end para a resposta de rota, a menos que você configure uma resposta de rota. Para saber mais sobre como configurar uma resposta de rota, consulte [the section called “Configurar respostas de rota da API do WebSocket”](#).

## Respostas de integração para comunicação bidirecional

As integrações podem ser divididas em integrações de proxy e não proxy.

### Important

Para integrações de proxy, o API Gateway transmite automaticamente a saída de backend para o autor da chamada como a carga completa. Não há uma resposta de integração.

Para integrações não proxy, você deve configurar ao menos uma resposta de integração:

- Idealmente, uma de suas respostas de integração deve agir como um genérico quando nenhuma escolha explícita puder ser feita. Este caso padrão é representado pela configuração de uma chave de resposta de integração `$default`.
- Em todos os outros casos, a chave de resposta de integração funciona como uma expressão regular. É necessário que siga um formato de `"/expression/"`.

Para integrações HTTP não proxy:

- O API Gateway tentará corresponder o código de status HTTP da resposta do backend. A chave de resposta de integração funcionará como uma expressão regular neste caso. Se não for possível encontrar uma correspondência, `$default` é escolhido como a resposta de integração.
- A expressão de seleção de modelo, conforme descrito acima, funciona de forma idêntica. Por exemplo:
  - `/2\d\d/`: Recebe e transforma respostas bem-sucedidas
  - `/4\d\d/`: Recebe e transforma erros de solicitação incorreta
  - `$default`: Recebe e transforma todas as respostas inesperadas

Para ter mais informações sobre expressões de seleção de modelo, consulte [the section called “Expressões de seleção de modelo”](#).

Configurar uma resposta de integração usando o console do API Gateway

Como configurar uma resposta de integração de rotas para uma API WebSocket usando o console do API Gateway:

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione sua API de WebSocket e rota.
3. Selecione a guia Solicitação de integração e, na seção Configurações de resposta de integração, selecione Criar resposta de integração.
4. Em Chave de resposta, insira um valor que será encontrado na chave de resposta na mensagem de saída após a avaliação da expressão de seleção de resposta. Por exemplo, é possível inserir `/4\d\d/` para receber e transformar erros de solicitação inválida ou inserir `$default` para receber e transformar todas as respostas que correspondam à expressão de seleção do modelo.
5. Em Expressão de seleção de modelos, insira uma expressão de seleção para avaliar a mensagem de saída.
6. Selecione Criar resposta.
7. Você também pode definir um modelo de mapeamento para configurar as transformações da carga útil da mensagem retornada. Selecione Criar modelo.
8. Insira um nome de chave. Se você estiver selecionando a expressão de seleção de modelo padrão, insira `\$default`.

9. Em Modelo de resposta, insira o modelo de mapeamento no editor de código.
10. Selecione Criar modelo.
11. Selecione Implantar API para implantar a API.

Use o comando [wscat](#) a seguir para se conectar à sua API. Para obter mais informações sobre o wscat, consulte [the section called “Use wscat para se conectar a uma API do WebSocket e enviar mensagens a ela”](#).

```
wscat -c wss://api-id.execute-api.us-east-2.amazonaws.com/test
```

Quando você chamar sua rota, a carga útil da mensagem retornada deverá ser exibida.

Configurar uma resposta de integração usando a AWS CLI

O comando [create-integration-response](#) indicado abaixo cria uma resposta de integração `$default`:

```
aws apigatewayv2 create-integration-response \
 --api-id vaz7da96z6 \
 --integration-id a1b2c3 \
 --integration-response-key '$default'
```

## Solicitar validação para APIs de WebSocket no API Gateway

É possível configurar o API Gateway para realizar a validação em uma solicitação de rota antes de prosseguir com a solicitação de integração. Se a validação falhar, o API Gateway falhará na solicitação sem chamar seu backend, enviará uma resposta de gateway “Corpo de solicitação incorreto” ao cliente e publicará os resultados da validação no CloudWatch Logs. Usar a validação dessa forma reduz chamadas desnecessárias para o backend da API.

### Expressões de seleção de modelo

Você pode usar uma expressão de seleção de modelo para validar solicitações dinamicamente dentro da mesma rota. A validação de modelos ocorrerá se você fornecer uma expressão de seleção de modelo para integrações proxy ou não proxy. Talvez seja necessário definir o modelo `$default` como um fallback quando nenhum modelo correspondente for encontrado. Se não houver nenhum modelo correspondente e `$default` não estiver definido, a validação falhará. A expressão de seleção é semelhante a `Route.ModelSelectionExpression` e é avaliada como a chave de `Route.RequestModels`.

Ao definir uma rota para uma API de WebSocket, é possível especificar, opcionalmente, uma expressão de seleção de modelo. Esta expressão é avaliada para selecionar o modelo a ser usado para a validação do corpo quando uma solicitação é recebida. A expressão é avaliada como uma das entradas em de uma rota [requestmodels](#).

Um modelo é expresso como um [esquema JSON](#) e descreve a estrutura de dados do corpo da solicitação. A natureza dessa expressão de seleção permite que você escolha dinamicamente o modelo para validar com base no tempo de execução para uma rota específica. Para obter informações sobre como criar um modelo, consulte [the section called “Modelos de dados para APIs REST”](#).

## Configurar a validação de solicitação usando o console do API Gateway

O seguinte exemplo mostra como configurar a validação da solicitação em uma rota.

Primeiro, crie um modelo e depois crie uma rota. Em seguida, configure a validação da solicitação na rota criada. Por fim, implante e teste a API. Para concluir este tutorial, você precisa de uma API de WebSocket com `$request.body.action` como a expressão de seleção de rota e um endpoint de integração para sua nova rota.

Você também precisa que o `wscat` se conecte à sua API. Para obter mais informações, consulte [the section called “Use wscat para se conectar a uma API do WebSocket e enviar mensagens a ela”](#).

### Como criar um modelo

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API de WebSocket.
3. No painel de navegação principal, selecione Modelos.
4. Escolha Criar modelo.
5. Em Nome, digite **emailModel**.
6. Em Tipo de conteúdo, insira **application/json**.
7. Em Esquema do modelo, insira o seguinte modelo:

```
{
 "$schema": "http://json-schema.org/draft-04/schema#",
 "type": "object",
 "required": ["address"],
 "properties": {
```

```
 "address": {
 "type": "string"
 }
 }
}
```

Este modelo exige que a solicitação contenha um endereço de e-mail.

## 8. Escolha Salvar.

Nesta etapa, você cria uma rota para a API de WebSocket.

Para criar uma rota

1. No painel de navegação principal, selecione Rotas.
2. Escolha Create route (Criar rota).
3. Em Route key (Chave de rota), insira **sendMessage**.
4. Escolha um tipo de integração e especifique um endpoint de integração. Para obter mais informações, consulte [the section called “Integrações”](#).
5. Escolha Create route (Criar rota).

Nesta etapa, você configura a validação da solicitação para a rota sendMessage.

Para configurar a validação da solicitação

1. Na guia Solicitação de rota, em Configurações de solicitação de rota, escolha Editar.
2. Em Expressão de seleção de modelos, insira **`${request.body.messageType}`**.

O API Gateway usa a propriedade messageType para validar a solicitação de entrada.

3. Escolha Adicionar modelo de solicitação.
4. Em Chave do modelo, insira **email**.
5. Em Modelo, escolha emailModel.

O API Gateway valida as mensagens de entrada com a propriedade messageType definida como email em relação a este modelo.



 Note

Se o API Gateway não combinar a expressão de seleção de modelo com uma chave do modelo, ele selecionará o modelo `$default`. Se não houver nenhum modelo `$default`, a validação falhará. Para APIs de produção, recomendamos que você crie um modelo `$default`.

## 6. Escolha Salvar alterações.

Nesta etapa, você implanta e testa a API.

Para implantar e testar sua API

1. Escolha Implantar API.
2. Escolha o estágio desejado na lista suspensa ou insira o nome de um novo estágio.
3. Escolha Implantar.
4. No painel de navegação principal, selecione Estágios.
5. Copie o URL WebSocket da API. O URL deve ser semelhante a `wss://abcdef123.execute-api.us-east-2.amazonaws.com/production`.
6. Abra um novo terminal e execute o comando `wscat` com os parâmetros a seguir.

```
wscat -c wss://abcdef123.execute-api.us-west-2.amazonaws.com/production
```

```
Connected (press CTRL+C to quit)
```

## 7. Use o comando a seguir para testar a API.

```
{"action": "sendMessage", "messageType": "email"}
```

```
{"message": "Invalid request body", "connectionId": "ABCD1=234",
 "requestId": "EFGH="}
```

O API Gateway falhará na solicitação.

Use o próximo comando para enviar uma solicitação válida para sua API.

```
{"action": "sendMessage", "messageType": "email", "address":
 "mary_major@example.com"}
```

## Transformações de dados para APIs de WebSocket no API Gateway

No API Gateway, a solicitação de método de uma API WebSocket pode usar uma carga em um formato diferente da carga da solicitação de integração correspondente, conforme exigido no backend. De maneira semelhante, o backend pode retornar uma carga de resposta de integração diferente da carga da resposta do método, conforme esperado pelo front-end.

O API Gateway permite usar transformações de modelos de mapeamento para mapear a carga útil de uma solicitação de método para a solicitação de integração correspondente ou de uma resposta de integração para a resposta de método correspondente. Você cria um modelo de mapeamento e especifica uma expressão de seleção de modelos para determinar qual deles usar para executar as transformações de dados necessárias.

É possível usar mapeamentos de dados para mapear dados de uma [solicitação de rota](#) para uma integração de backend. Para saber mais, consulte [the section called “Mapeamento de dados”](#).

### Modelos e modelos de mapeamento

Um modelo de mapeamento é um script expresso em [Velocity Template Language \(VTL\)](#) e aplicado à carga usando [expressões JSONPath](#). Para obter mais informações sobre modelos de mapeamento do API Gateway, consulte [Transformações de modelo de mapeamento para APIs REST no API Gateway](#).

A carga pode ter um modelo de dados de acordo com o [esquema JSON rascunho 4](#). Não é necessário definir um modelo para criar um modelo de mapeamento. No entanto, um modelo pode ajudar você a criar um modelo, porque o API Gateway gera um esquema de modelo com base em um modelo fornecido. Para obter mais informações sobre modelos do API Gateway, consulte [Modelos de dados para APIs REST](#).

### Expressões de seleção de modelo

Para transformar uma carga com um modelo de mapeamento, especifique uma expressão de seleção de modelo de API WebSocket em uma [solicitação de integração](#) ou [resposta de integração](#). Esta expressão é avaliada para determinar o modelo de entrada ou de saída (se houver) a ser

utilizado para transformar o corpo da solicitação no corpo da solicitação de integração (por meio de um modelo de entrada) ou o corpo da resposta de integração para o corpo de resposta de rotas (por meio de um modelo de saída).

`Integration.TemplateSelectionExpression` oferece suporte a `${request.body.jsonPath}` e valores estáticos.

`IntegrationResponse.TemplateSelectionExpression` oferece suporte a `${request.body.jsonPath}`, `${integration.response.statuscode}`, `${integration.response.header.headerName}`, `${integration.response.multivalueheader.headerName}` e a valores estáticos.

## Expressões de seleção de resposta da integração

Quando você [define uma resposta de integração](#) para uma API WebSocket, é possível especificar opcionalmente uma expressão de seleção de resposta da integração. Esta expressão determina qual [IntegrationResponse](#) deve ser selecionada quando uma integração retorna. O valor dessa expressão é atualmente restrito pelo API Gateway, conforme definido abaixo. Observe que essa expressão só é relevante para integrações não proxy; uma integração de proxy simplesmente transmitirá a carga da resposta de volta para o autor da chamada sem modelagem ou modificação.

Ao contrário de outras expressões de seleção anteriores, essa expressão é atualmente compatível com um formato de padrão correspondente. A expressão deve ser encapsulada com barras.

Atualmente, o valor é corrigido dependendo do [integrationType](#):

- Para integrações baseadas em Lambda, é `$integration.response.body.errorMessage`.
- Para integrações HTTP e MOCK, é `$integration.response.statuscode`.
- Para HTTP\_PROXY e AWS\_PROXY, a expressão não é utilizada porque você está solicitando que a carga seja transmitida para o autor da chamada.

## Configurar o mapeamento de dados para APIs de WebSocket no API Gateway

O mapeamento de dados permite mapear dados de uma [solicitação de rota](#) para uma integração de backend.

**Note**

O mapeamento de dados para APIs do WebSocket não é suportado no Console de gerenciamento da AWS. É necessário usar a AWS CLI, o AWS CloudFormation, ou um SDK para configurar o mapeamento de dados.

**Tópicos**

- [Mapear dados de solicitação de rota para parâmetros de solicitação de integração](#)
- [Exemplos](#)

**Mapear dados de solicitação de rota para parâmetros de solicitação de integração**

Os parâmetros de solicitação de integração podem ser mapeados a partir de quaisquer parâmetros de solicitação de rota definidos, o corpo da solicitação, as variáveis [context](#) ou [stage](#) e valores estáticos.

A tabela a seguir mostra as expressões de mapeamento de dados da solicitação de integração. Na tabela, *PARAM\_NAME* é o nome de um parâmetro de solicitação de rota do tipo de parâmetro especificado. Deve corresponder à expressão regular `^[a-zA-Z0-9._$-]+$`. *JSONPath\_Expression* é uma expressão JSONPath para um campo JSON do corpo da solicitação.

| Fonte de dados mapeada                                                                       | Expressão de mapeamento                                              |
|----------------------------------------------------------------------------------------------|----------------------------------------------------------------------|
| String de consulta de solicitação (compatível apenas com a rota \$connect)                   | <code>route.request.querystring. <i>PARAM_NAME</i></code>            |
| Cabeçalho de solicitação (compatível apenas com a rota \$connect)                            | <code>route.request.header. <i>PARAM_NAME</i></code>                 |
| String de consulta de solicitação de vários valores (compatível apenas com a rota \$connect) | <code>route.request.multivaluedquerystring. <i>PARAM_NAME</i></code> |
| Cabeçalho de solicitação de vários valores (compatível apenas com a rota \$connect)          | <code>route.request.multivaluedheader. <i>PARAM_NAME</i></code>      |

| Fonte de dados mapeada | Expressão de mapeamento                                                                                             |
|------------------------|---------------------------------------------------------------------------------------------------------------------|
| Corpo da solicitação   | <code>route.request.body.</code> <i>JSONPath_EXPRESSION</i>                                                         |
| Variáveis de estágio   | <code>stageVariables.</code> <i>VARIABLE_NAME</i>                                                                   |
| Variáveis de contexto  | <code>context.</code> <i>VARIABLE_NAME</i> que deve ser uma das <a href="#">variáveis de contexto com suporte</a> . |
| Valor estático         | <i>'STATIC_VALUE'</i> . <i>STATIC_VALUE</i> é um literal de string e deve estar entre aspas simples.                |

Ao criar um mapeamento de dados, usando a AWS CLI, siga o formato correto para usar literais com strings na AWS CLI. Para ter mais informações, consulte [Usar aspas e literais com strings na AWS CLI](#) no Guia do usuário da versão 1 da AWS Command Line Interface.

## Exemplos

Os exemplos da AWS CLI a seguir configuram mapeamentos de dados. Para obter um modelo demonstrativo do CloudFormation, consulte [websocket-data-mapping.yaml](#).

Mapear o `connectionId` de um cliente para um cabeçalho em uma solicitação de integração

O comando [update-integration](#) indicado abaixo associa o `connectionId` de um cliente a um cabeçalho `connectionId` na solicitação para uma integração de backend:

```
aws apigatewayv2 update-integration \
 --integration-id abc123 \
 --api-id a1b2c3d4 \
 --request-parameters
 'integration.request.header.connectionId'='context.connectionId'
```

Mapear um parâmetro de string de consulta para um cabeçalho em uma solicitação de integração

O exemplo a seguir associa um parâmetro de string de consulta `authToken` a um cabeçalho `authToken` na solicitação de integração.

1. Use o comando [update-route](#) a seguir para adicionar o parâmetro de string de consulta authToken aos parâmetros de solicitação da rota.

```
aws apigatewayv2 update-route --route-id 0abcdef \
 --api-id a1b2c3d4 \
 --request-parameters '{"route.request.querystring.authToken": {"Required":
false}}'
```

2. Use o comando [update-integration](#) a seguir para associar o parâmetro de string de consulta ao cabeçalho authToken na solicitação para a integração de backend.

```
aws apigatewayv2 update-integration \
 --integration-id abc123 \
 --api-id a1b2c3d4 \
 --request-parameters
'integration.request.header.authToken='route.request.querystring.authToken'
```

3. (Opcional) Se necessário, use o comando [delete-route-request-parameter](#) a seguir para excluir o parâmetro de string de consulta authToken dos parâmetros de solicitação da rota.

```
aws apigatewayv2 delete-route-request-parameter \
 --route-id 0abcdef \
 --api-id a1b2c3d4 \
 --request-parameter-key 'route.request.querystring.authToken'
```

## Referência de modelos de mapeamento da API de WebSocket do API Gateway

Esta seção resume o conjunto de variáveis que são atualmente compatíveis com APIs WebSocket no API Gateway.

| Parâmetro                           | Descrição                                                                                |
|-------------------------------------|------------------------------------------------------------------------------------------|
| <code>\$context.connectionId</code> | Um ID exclusivo para a conexão que pode ser utilizado para fazer uma chamada ao cliente. |
| <code>\$context.connectedAt</code>  | O tempo de conexão formatado em <a href="#">Epoch</a> .                                  |

| Parâmetro                                     | Descrição                                                                                                                                                                                                           |
|-----------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$context.domainName</code>             | Um nome de domínio para a API WebSocket. É possível utilizá-lo para fazer uma chamada ao cliente (em vez de um valor codificado).                                                                                   |
| <code>\$context.eventType</code>              | O tipo de evento: <code>CONNECT</code> , <code>MESSAGE</code> ou <code>DISCONNECT</code> .                                                                                                                          |
| <code>\$context.messageId</code>              | Um ID exclusivo do servidor para uma mensagem. Disponível apenas quando o <code>\$context.eventType</code> é <code>MESSAGE</code> .                                                                                 |
| <code>\$context.routeKey</code>               | A chave de roteamento selecionada.                                                                                                                                                                                  |
| <code>\$context.requestId</code>              | Igual a <code>\$context.extendedRequestId</code> .                                                                                                                                                                  |
| <code>\$context.extendedRequestId</code>      | Um ID gerado automaticamente para a chamada de API, que contém mais informações úteis para depuração/solução de problemas.                                                                                          |
| <code>\$context.apiId</code>                  | O identificador que o API Gateway atribui à sua API.                                                                                                                                                                |
| <code>\$context.authorizer.principalId</code> | A identificação do usuário principal associada ao token enviado pelo cliente e retornada por uma função do Lambda do autorizador do Lambda do API Gateway (anteriormente conhecido como autorizador personalizado). |

| Parâmetro                                          | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|----------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$context.authorizer.</code> <i>property</i> | <p>O valor transformado em string do par de chave/valor especificado do mapa context retornado de uma função de autorizador do Lambda do API Gateway. Por exemplo, se o autorizador retornar o seguinte mapa context:</p> <pre>"context" : {<br/>  "key": "value",<br/>  "numKey": 1,<br/>  "boolKey": true<br/>}</pre> <p>chamar <code>\$context.authorizer.key</code> retornará a string "value", chamar <code>\$context.authorizer.numKey</code> retornará a string "1" e chamar <code>\$context.authorizer.boolKey</code> retornará a string "true".</p> |
| <code>\$context.error.messageString</code>         | O valor citado de <code>\$context.error.message</code> , ou seja, " <code>\$context.error.message</code> ".                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <code>\$context.error.validationErrorString</code> | Uma string que contém uma mensagem de erro de validação detalhada.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <code>\$context.identity.accountId</code>          | O ID da conta da AWS associado à solicitação.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <code>\$context.identity.apiKey</code>             | A chave do proprietário da API associada à solicitação de API habilitada por chave.                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <code>\$context.identity.apiKeyId</code>           | O ID da chave da API associada à solicitação de API habilitada por chave                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <code>\$context.identity.caller</code>             | O identificador principal do agente de chamada que está fazendo a solicitação.                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |



| Parâmetro                                                     | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|---------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$context.identity.cognitoAuthenticationProvider</code> | <p>Uma lista separada por vírgulas de todos os provedores de autenticação do Amazon Cognito usados pelo autor da chamada que faz a solicitação. Disponível somente se a solicitação foi assinada com credenciais do Amazon Cognito.</p> <p>Por exemplo, para uma identidade e de um grupo de usuários do Amazon Cognito, <code>cognito-idp.<i>region</i>.amazonaws.com/<i>user_pool_id</i></code>, <code>cognito-idp.<i>region</i>.amazonaws.com/<i>user_pool_id</i>:CognitoSignIn:<i>token subject claim</i></code></p> <p>Consulte informações sobre os provedores de autenticação do Amazon Cognito disponíveis em <a href="#">Using Federated Identities</a> no Guia do desenvolvedor do Amazon Cognito.</p> |
| <code>\$context.identity.cognitoAuthenticationType</code>     | <p>O tipo de autenticação do Amazon Cognito do autor da chamada que faz a solicitação. Disponível somente se a solicitação foi assinada com credenciais do Amazon Cognito. Os valores possíveis incluem <code>authenticated</code> para identidades autenticadas e <code>unauthenticated</code> para identidades não autenticadas.</p>                                                                                                                                                                                                                                                                                                                                                                           |
| <code>\$context.identity.cognitoIdentityId</code>             | <p>O ID de identidade do Amazon Cognito do autor da chamada que faz a solicitação. Disponível somente se a solicitação foi assinada com credenciais do Amazon Cognito.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

| Parâmetro                                                              | Descrição                                                                                                                                                                     |
|------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$context.identity.cognitoId</code><br><code>entityPoolId</code> | O ID do grupo de identidades do Amazon Cognito do autor da chamada que faz a solicitação. Disponível somente se a solicitação foi assinada com credenciais do Amazon Cognito. |
| <code>\$context.identity.sourceIp</code>                               | O endereço IP de origem da conexão TCP mais próxima que está fazendo a solicitação para o endpoint do API Gateway.                                                            |
| <code>\$context.identity.user</code>                                   | O identificador principal do usuário que está fazendo a solicitação.                                                                                                          |
| <code>\$context.identity.userAgent</code>                              | O agente de usuário do autor da chamada da API.                                                                                                                               |
| <code>\$context.identity.userArn</code>                                | O Nome do Recurso Amazon (ARN) do usuário efetivo identificado após a autenticação.                                                                                           |
| <code>\$context.requestTime</code>                                     | O horário da solicitação <a href="#">CLF</a> formatado (dd/MMM/yyyy:HH:mm:ss +-hhmm ).                                                                                        |
| <code>\$context.requestTimeEpoch</code>                                | O tempo de solicitação formatado em <a href="#">Epoch</a> , em milissegundos.                                                                                                 |
| <code>\$context.stage</code>                                           | O estágio de implantação da chamada de API (por exemplo, Beta ou Prod).                                                                                                       |
| <code>\$context.status</code>                                          | O status da resposta.                                                                                                                                                         |
| <code>\$input.body</code>                                              | Retorna a carga bruta como uma string.                                                                                                                                        |

| Parâmetro                    | Descrição                                                                                                                                                                                                                                                                                                                                                           |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$input.json(x)</code> | <p>Essa função avalia uma expressão JSONPath e retorna os resultados como uma string JSON.</p> <p>Por exemplo, <code>\$input.json('\$.pets')</code> retornará uma string JSON que representa a estrutura de animais de estimação.</p> <p>Para obter mais informações sobre o JSONPath, consulte <a href="#">JSONPath</a> ou <a href="#">JSONPath para Java</a>.</p> |

| Parâmetro                                            | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$input.path(x)</code>                         | <p>Usa uma string de expressão JSONPath (x) e retorna uma representação de objeto JSON do resultado. Isso permite que você acesse e manipule elementos da carga nativamente em <a href="#">Apache Velocity Template Language (VTL)</a>.</p> <p>Por exemplo, se a expressão <code>\$input.path( '\$.pets' )</code> retorna um objeto da seguinte forma:</p> <pre>[   {     "id": 1,     "type": "dog",     "price": 249.99   },   {     "id": 2,     "type": "cat",     "price": 124.99   },   {     "id": 3,     "type": "fish",     "price": 0.99   } ]</pre> <p><code>\$input.path( '\$.pets' ).count()</code> retornaria "3".</p> <p>Para obter mais informações sobre o JSONPath, consulte <a href="#">JSONPath</a> ou <a href="#">JSONPath para Java</a>.</p> |
| <code>\$stageVariables. &lt;variable_name&gt;</code> | <p><code>&lt;variable_name&gt;</code> representa um nome de variável de estágio.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

| Parâmetro                                                 | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-----------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$stageVariables[' &lt;variable_name&gt; ']</code>  | <code>&lt;variable_name&gt;</code> representa qualquer nome de variável de estágio.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <code>\${stageVariables[' &lt;variable_name&gt;']}</code> | <code>&lt;variable_name&gt;</code> representa qualquer nome de variável de estágio.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <code>\$util.escapeJavaScript()</code>                    | <p>Escapa os caracteres em uma string usando regras de string JavaScript.</p> <div><p> <b>Note</b></p><p>Essa função transformará quaisquer aspas simples comuns ( ' ) em aspas com escape ( \ ' ). No entanto, as aspas simples com escape não são válidas no JSON. Portanto, quando a saída dessa função for usada em uma propriedade JSON, você deverá transformar todas aspas simples ( \ ' ) com escape de volta para aspas simples comuns ( ' ). Isso é mostrado no exemplo a seguir:</p><pre>\$util.escapeJavaScript(<br/>  ript( <i>data</i> ).replaceAll("\\'",<br/>    ,"'")</pre></div> |

| Parâmetro                          | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$util.parseJson()</code>    | <p>Usa um JSON "transformado em string" e retorna uma representação de objeto do resultado. Você pode usar o resultado dessa função para acessar e manipular elementos da carga nativamente em Apache VTL (Velocity Template Language). Por exemplo, se tiver a seguinte carga:</p> <pre>{"errorMessage":{"key1":"var1","key2":{"arr":[1,2,3]}}}</pre> <p>e usar o seguinte modelo de mapeamento</p> <pre>#set (\$errorMessageObj = \$util.parseJson(\$input.path('\$errorMessage')))<br/>{<br/>    "errorMessageObjKey2ArrVal" :<br/>    \$errorMessageObj.key2.arr[0]<br/>}</pre> <p>Você receberá a seguinte saída:</p> <pre>{<br/>    "errorMessageObjKey2ArrVal" : 1<br/>}</pre> |
| <code>\$util.urlEncode()</code>    | Converte uma string no formato "application/x-www-form-urlencoded".                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <code>\$util.urlDecode()</code>    | Decodifica uma string "application/x-www-form-urlencoded".                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <code>\$util.base64Encode()</code> | Codifica os dados em uma string codificada em base64.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

| Parâmetro                          | Descrição                                               |
|------------------------------------|---------------------------------------------------------|
| <code>\$util.base64Decode()</code> | Decodifica os dados de uma string codificada em base64. |

## Tipos de mídia binários para APIs de WebSocket no API Gateway

As APIs WebSocket do API Gateway não são compatíveis com quadros binários nas cargas de mensagem recebida. Se um aplicativo do cliente enviar um quadro binário, este será rejeitado pelo API Gateway, que desconectará o cliente com o código 1003.

Há uma solução para esse comportamento. Se o cliente envia dados binários codificados em texto (por exemplo, base64) como um quadro de texto, você pode definir a propriedade `contentHandlingStrategy` da integração como `CONVERT_TO_BINARY` para converter a carga de string codificada em base64 em binário.

Para retornar uma resposta de rota para uma carga de binário em integrações não proxy, você pode definir a propriedade `contentHandlingStrategy` da resposta de integração como `CONVERT_TO_TEXT` para converter a carga de binário em string codificada em base64.

## Invocar APIs de WebSocket

Depois de implantar a API WebSocket, as aplicações cliente poderão se conectar para enviar mensagens para ela, e seu serviço de backend poderá enviar mensagens para aplicações cliente conectados:

- Você pode usar `wscat` para se conectar à sua API WebSocket e enviar mensagens a ela, simulando o comportamento do cliente. Consulte [the section called “Use wscat para se conectar a uma API do WebSocket e enviar mensagens a ela”](#).
- Você pode usar a API `@connections` do seu serviço de backend para enviar uma mensagem de retorno a um cliente conectado, obter informações de conexão ou desconectar o cliente. Consulte [the section called “Usar os comandos @connections em seu serviço de backend”](#).
- Um aplicativo do cliente pode usar sua própria biblioteca WebSocket para invocar a API WebSocket.

## Use **wscat** para se conectar a uma API do WebSocket e enviar mensagens a ela

O utilitário [wscat](#) é uma ferramenta conveniente para testar uma API WebSocket criada e implantada no API Gateway. Você pode instalar e utilizar wscat da seguinte maneira:

1. Faça download de wscat em <https://www.npmjs.com/package/wscat>.
2. Instale o wscat executando o seguinte comando.

```
npm install -g wscat
```

3. Para se conectar à sua API, execute o comando wscat conforme mostrado no exemplo a seguir. Observe que este exemplo pressupõe que a configuração Authorization é NONE.

```
wscat -c wss://aabbccdde.execute-api.us-east-1.amazonaws.com/test/
```

É necessário substituir *aabbccdde* pelo ID de API real que é exibido no console do API Gateway ou retornado pelo comando [create-api](#) da AWS CLI.

Além disso, se a sua API estiver em uma região diferente de *us-east-1*, será necessário substituir a região correta.

4. Enquanto estiver conectado, para testar a API, insira uma mensagem como a seguinte:

```
{"{jsonpath-expression}":"{route-key}"}
```

em que *{jsonpath-expression}* é uma expressão JSONPath e *{route-key}* é uma chave de roteamento para a API. Por exemplo:

```
{"action":"action1"}
{"message":"test response body"}
```

Para obter mais informações sobre o JSONPath, consulte [JSONPath](#) ou [JSONPath para Java](#).

5. Para desconectar-se da API, insira `ctrl-C`.



## Usar os comandos **@connections** em seu serviço de backend

Seu serviço de backend pode utilizar as solicitações HTTP de conexão WebSocket a seguir para enviar uma mensagem de retorno a um cliente conectado, obter informações de conexão ou desconectar o cliente.

### Important

Essas solicitações utilizam a [autorização do IAM](#), portanto, você deve assiná-las com o [Signature Version 4 \(SigV4\)](#). Para fazer isso, você pode usar a API de Gerenciamento do API Gateway. Para obter mais informações, consulte [ApiGatewayManagementApi](#).

No comando a seguir, é necessário substituir **{api-id}** pelo ID de API real que é exibido no console do API Gateway ou retornado pelo comando [create-api](#) da AWS CLI. É necessário estabelecer conexão antes de usar esse comando.

Para enviar uma mensagem de retorno ao cliente, utilize:

```
POST https://{api-id}.execute-api.us-east-1.amazonaws.com/{stage}/@connections/{connection_id}
```

É possível testar essa solicitação usando [Postman](#) ou chamando [awscurl](#) conforme o exemplo a seguir:

```
awscurl --service execute-api -X POST -d "hello world" https://{prefix}.execute-api.us-east-1.amazonaws.com/{stage}/@connections/{connection_id}
```

Você precisa codificar o URL no comando, conforme mostrado no exemplo a seguir:

```
awscurl --service execute-api -X POST -d "hello world" https://aabbccdde.execute-api.us-east-1.amazonaws.com/prod/%40connections/R0oXAdFD0kwCH6w%3D
```

Para obter o status de conexão mais recente do cliente, use:

```
GET https://{api-id}.execute-api.us-east-1.amazonaws.com/{stage}/@connections/{connection_id}
```

Para desconectar o cliente, use:

```
DELETE https://{api-id}.execute-api.us-east-1.amazonaws.com/{stage}/
@connections/{connection_id}
```

Você pode criar dinamicamente uma URL de retorno de chamada ao utilizar as variáveis `$context` em sua integração. Por exemplo, se usar a integração de proxy do Lambda com uma função do Lambda Node.js, você poderá criar o URL e enviar uma mensagem para um cliente conectado da seguinte forma:

```
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,
 };
};
```

Se você usar um nome de domínio personalizado para sua API de WebSocket, remova a variável `stage` do código da função.

Ao enviar uma mensagem de retorno de chamada, a função do Lambda deve ter permissão para chamar a API de gerenciamento do API Gateway. Você poderá receber um erro contendo `GoneException` se publicar uma mensagem antes do estabelecimento da conexão ou após a desconexão do cliente.

## Publicar APIs de WebSocket para os clientes invocarem

Apenas criar e desenvolver uma API do API Gateway não fará com que ela possa ser chamada automaticamente pelos usuários. Para que ela possa ser chamada, implante a API em um estágio. Além disso, você pode querer personalizar o URL que os usuários usarão para acessar a API. Você pode dar a ele um domínio que seja consistente com sua marca ou que seja mais memorável que o URL padrão da API.

Nesta seção, você pode aprender a implantar a API e personalizar o URL fornecido aos usuários para acessá-la.

### Note

Para aumentar a segurança das APIs do API Gateway, o domínio execute-`api`.`{region}`.amazonaws.com é registrado na [Public Suffix List \(PSL\)](#). Para maior segurança, recomendamos que você use cookies com um prefixo `__Host-` se precisar definir cookies confidenciais no nome de domínio padrão para as APIs do API Gateway. Essa prática ajudará a defender seu domínio contra tentativas de falsificação de solicitação entre sites (CSRF). Para obter mais informações, consulte a página [Set-Cookie](#) na Mozilla Developer Network.

### Tópicos

- [Criar estágios para APIs de WebSocket no API Gateway](#)
- [Implantar APIs de WebSocket no API Gateway](#)
- [Política de segurança para APIs de WebSocket no API Gateway](#)
- [Nomes de domínio personalizados para APIs de WebSocket no API Gateway](#)

## Criar estágios para APIs de WebSocket no API Gateway

O estágio de uma API é uma referência lógica a um estado do ciclo de vida de sua API (por exemplo, dev, prod, beta ou v2). Os estágios de API são identificados por seu ID de API e nome de estágio

e são incluídos no URL que você usa para chamar a API. Cada estágio é uma referência nomeada a uma implantação da API e é disponibilizado para chamadas feitas por aplicativos cliente.

Uma implantação é um instantâneo da configuração da API. Depois de implantar uma API em um estágio, ela estará disponível para ser chamada por clientes. Você deve implantar uma API para que as alterações entrem em vigor.

## Variáveis de estágio

As variáveis de estágio são pares chave-valor que você pode definir para um estágio de uma API WebSocket. Elas atuam como variáveis de ambiente e podem ser usadas na configuração da API.

Por exemplo, você pode definir uma variável de estágio e, depois, definir seu valor como um endpoint HTTP para uma integração de proxy HTTP. Posteriormente, você pode fazer referência ao endpoint usando o nome da variável de estágio associada. Fazendo isso, você pode usar a mesma configuração de API com um endpoint diferente em cada estágio. Da mesma forma, você pode usar variáveis de estágio para especificar uma integração de função diferente do AWS Lambda para cada estágio da API.

### Note

As variáveis de estágio não se destinam a ser usadas para dados confidenciais, como credenciais. Para transmitir dados confidenciais para integrações, use um autorizador do AWS Lambda. Você pode passar dados confidenciais para integrações na saída do autorizador do Lambda. Para saber mais, consulte [the section called “Formato de resposta do autorizador do Lambda”](#).

## Exemplos

Para usar uma variável de estágio para personalizar o endpoint de integração HTTP, primeiro defina o nome e o valor da variável de estágio (por exemplo, `url`) com um valor de `example.com`. Depois, configure uma integração de proxy HTTP. Em vez de inserir o URL do endpoint, você pode instruir o API Gateway a usar o valor da variável de estágio, **`http://{stageVariables.url}`**. Esse valor instrui o API Gateway a substituir sua variável de estágio `{}` em tempo de execução, dependendo do estágio da API.

É possível fazer referência a variáveis de estágio de forma semelhante para especificar um nome de função do Lambda ou um ARN de função da AWS.

Ao especificar um nome de função do Lambda como um valor de variável de estágio, você deve configurar as permissões nessa função do Lambda manualmente. O comando [add-permission](#) indicado abaixo adiciona as permissões necessárias:

```
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
```

## Referência de variáveis de estágio do API Gateway

### URIs de integração HTTP

Uma variável de estágio pode ser usada como parte de um URI de integração HTTP, como mostram os exemplos a seguir.

- Um URI completo sem protocolo – `http://${stageVariables.<variable_name>}`
- Um domínio complet – `http://${stageVariables.<variable_name>}/resource/operation`
- Um subdomíni – `http://${stageVariables.<variable_name>}.example.com/resource/operation`
- Um caminh – `http://example.com/${stageVariables.<variable_name>}/bar`
- Uma string de consult – `http://example.com/foo?q=${stageVariables.<variable_name>}`

### Funções do Lambda

É possível usar uma variável de estágio no lugar de um nome de função ou alias do Lambda, conforme mostrado nos exemplos a seguir.

- `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 usar uma variável de estágio para uma função do Lambda, a função deve estar na mesma conta que a API. As variáveis de estágio não suportam funções do Lambda entre contas.

## AWSCredenciais de integração da

É possível usar uma variável de estágio como parte de um ARN de credencial de usuário ou de função da AWS, conforme mostrado no exemplo a seguir.

- `arn:aws:iam::<account_id>:${stageVariables.<variable_name>}`

## Implantar APIs de WebSocket no API Gateway

Depois de criar sua API WebSocket, você deve implantá-la para permitir que seja invocada por seus usuários após a disponibilização.

Para implantar uma API, você cria uma [implantação da API](#) e a associa a um [estágio](#). Cada estágio é um snapshot da API, sendo disponibilizado para ser chamado pelos aplicativos do cliente.

 Important

Toda vez que atualizar uma API, você deverá reimplantá-la. Alterações em qualquer item que não sejam as configurações do estágio exigem uma nova implantação, incluindo modificações nos seguintes recursos:

- Rotas
- Integrações
- Authorizers

Por padrão, você está limitado a dez estágios para cada API. Recomendamos reutilizar estágios para as implantações.

Para chamar uma API WebSocket implantada, o cliente envia uma mensagem à URL da API. A URL é determinada pelo nome do host e pelo nome do estágio da API.

 Note

O API Gateway oferecerá suporte a cargas de até 128 KB com quadros de máximo 32 KB. Se uma mensagem exceder 32 KB, deve ser dividida em vários quadros, cada qual contendo 32 KB ou menos.

Com o nome de domínio padrão da API, a URL base de uma (por exemplo) API WebSocket em determinado estágio (*{stageName}*) tem o seguinte formato:

```
wss://{api-id}.execute-api.{region}.amazonaws.com/{stageName}
```

Para facilitar o uso da URL da API WebSocket, você pode criar um nome de domínio personalizado (por exemplo, `api.example.com`) para substituir o nome de domínio padrão da API. O processo de configuração é o mesmo utilizado para APIs REST. Para obter mais informações, consulte [the section called “Nomes de domínios personalizados”](#).

Os estágios permitem um controle de versão robusto para sua API. Por exemplo, você pode implantar uma API em um estágio `test` e um `prod`, e usar o estágio `test` como uma compilação de teste e o estágio `prod` como uma compilação estável. Depois que as atualizações passarem no teste, você poderá promover o estágio `test` para o estágio `prod`. Essa promoção pode ser feita por meio da reimplantação da API ao estágio `prod`. Para obter mais detalhes sobre estágios, consulte [the section called “Configurar um estágio”](#).

## Tópicos

- [Criar uma implantação de API do WebSocket usando a AWS CLI](#)
- [Criar uma implantação de API WebSocket usando o console do API Gateway](#)

## Criar uma implantação de API do WebSocket usando a AWS CLI

O seguinte comando [create-deployment](#) cria uma implantação:

```
aws apigatewayv2 --region us-east-1 create-deployment --api-id aabbccdde
```

A saída será exibida da seguinte forma:

```
{
 "DeploymentId": "fedcba",
```

```
"DeploymentStatus": "DEPLOYED",
"CreateDate": "2018-11-15T06:49:09Z"
}
```

A API implantada não poderá ser chamada até que a implantação seja associada a um estágio. Você pode criar um novo estágio ou reutilizar um estágio que criou anteriormente.

O comando [create-stage](#) indicado abaixo cria um estágio e o associa à implantação:

```
aws apigatewayv2 --region us-east-1 create-stage --api-id aabbccdde --deployment-id
fedcba --stage-name test
```

A saída será exibida como a seguir:

```
{
 "StageName": "test",
 "CreateDate": "2018-11-15T06:50:28Z",
 "DeploymentId": "fedcba",
 "DefaultRouteSettings": {
 "MetricsEnabled": false,
 "ThrottlingBurstLimit": 5000,
 "DataTraceEnabled": false,
 "ThrottlingRateLimit": 10000.0
 },
 "LastUpdatedDate": "2018-11-15T06:50:28Z",
 "StageVariables": {},
 "RouteSettings": {}
}
```

É possível reutilizar um estágio existente atualizando a propriedade `deploymentId` do estágio com o ID da implantação recém-criado (*deployment-id*). O comando [update-stage](#) indicado abaixo atualiza o ID de implantação do estágio:

```
aws apigatewayv2 update-stage --region region \
 --api-id api-id \
 --stage-name stage-name \
 --deployment-id deployment-id
```

## Criar uma implantação de API WebSocket usando o console do API Gateway

Para usar o console do API Gateway a fim de criar uma implantação para uma API WebSocket:



1. Inicie uma sessão no console do API Gateway e escolha a API.
2. Escolha Implantar API.
3. Escolha o estágio desejado na lista suspensa ou insira o nome de um novo estágio.

## Política de segurança para APIs de WebSocket no API Gateway

O API Gateway impõe uma política de segurança de TLS\_1\_2 para todos os endpoints da API de WebSocket.

Uma política de segurança é uma combinação predefinida da versão mínima do TLS e dos pacotes de criptografia oferecida pelo Amazon API Gateway. O protocolo TLS trata problemas de segurança de rede, como violação e interceptação entre um cliente e o servidor. Quando seus clientes estabelecem um handshake do TLS para a API por meio do domínio personalizado, a política de segurança aplica as opções do pacote de criptografia e da versão do TLS que seus clientes podem optar por usar. Essa política de segurança é compatível com tráfego TLS 1.2 e TLS 1.3 e rejeita tráfego TLS 1.0.

### Protocolos e cifras TLS compatíveis para APIs de WebSocket

A tabela a seguir descreve os protocolos TLS compatíveis com as APIs de WebSocket.

| Protocolos TLS | Política de segurança TLS_1_2 |
|----------------|-------------------------------|
| TLSv1.3        | Sim                           |
| TLSv1.2        | Sim                           |

A tabela a seguir descreve as criptografias TLS disponíveis para a política de segurança TLS 1\_2 para APIs de WebSocket.

| Cifras TLS             | Política de segurança TLS_1_2 |
|------------------------|-------------------------------|
| TLS_AES_128_GCM_SHA256 | Sim                           |
| TLS_AES_256_GCM_SHA384 | Sim                           |

| Cifras TLS                    | Política de segurança TLS_1_2 |
|-------------------------------|-------------------------------|
| TLS_CHACHA20_POLY1305_SHA256  | Sim                           |
| ECDHE-ECDSA-AES128-GCM-SHA256 | Sim                           |
| ECDHE-RSA-AES128-GCM-SHA256   | Sim                           |
| ECDHE-ECDSA-AES128-SHA256     | Sim                           |
| ECDHE-RSA-AES128-SHA256       | Sim                           |
| ECDHE-ECDSA-AES256-GCM-SHA384 | Sim                           |
| ECDHE-RSA-AES256-GCM-SHA384   | Sim                           |
| ECDHE-ECDSA-AES256-SHA384     | Sim                           |
| ECDHE-RSA-AES256-SHA384       | Sim                           |
| AES128-GCM-SHA256             | Sim                           |
| AES128-SHA256                 | Sim                           |
| AES256-GCM-SHA384             | Sim                           |
| AES256-SHA256                 | Sim                           |

## Nomes das criptografias OpenSSL e RFC

OpenSSL e IETF RFC 5246 usam nomes diferentes para as mesmas cifras. Para ver uma lista dos nomes das cifras, consulte [the section called “Nomes das criptografias OpenSSL e RFC”](#).

## Informações sobre APIs HTTP e APIs REST

Para saber mais sobre APIs REST e APIs HTTP, consulte [the section called “Escolher uma política de segurança”](#) e [the section called “Política de segurança para APIs HTTP no API Gateway”](#).

# Nomes de domínio personalizados para APIs de WebSocket no API Gateway

Os nomes de domínio personalizados são URLs mais simples e intuitivos que você pode fornecer aos usuários da API.

Após a implantação da sua API, você e seus clientes podem invocar essa API usando a URL de base padrão com o seguinte formato:

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

em que *api-id* é gerado pelo API Gateway, *region* é a região da AWS e *stage* é especificado por você ao implantar a API.

A parte do nome de host do URL, *api-id*.execute-api.*region*.amazonaws.com, refere-se a um endpoint de API. O endpoint de API padrão é gerado aleatoriamente, difícil de lembrar e não é simples de usar.

Com nomes de domínio personalizados, você pode configurar o nome de host da API e escolher um caminho base (por exemplo, myservice) para mapear o URL alternativo para sua API. Por exemplo, um URL de base de API mais amigável pode se tornar:

```
https://api.example.com/myservice
```

## Considerações

As considerações a seguir podem afetar o uso de um nome de domínio personalizado.

- Se mapear um nome de domínio personalizado para uma API de WebSocket, você não poderá mapeá-lo para uma API REST ou uma API HTTP.
- Somente nomes de domínio regionais personalizados são compatíveis.
- Para a versão mínima de TLS, somente o TLS 1.2 é compatível.
- É necessário criar ou atualizar o registro de recursos do provedor DNS para ser mapeado ao endpoint da API. Sem esse mapeamento, as solicitações de API que forem direcionadas para o nome de domínio personalizado não conseguirão acessar o API Gateway.
- É possível comportar um número quase infinito de nomes de domínio sem exceder a cota padrão usando um certificado curinga. Para obter mais informações, consulte [the section called “Nomes de domínio personalizados curinga”](#).

## Pré-requisitos

Veja a seguir os pré-requisitos de um nome de domínio personalizado.

### Registrar um nome de domínio

É necessário ter um nome de domínio da Internet registrado para configurar nomes de domínio personalizados para as APIs. É possível registrar o domínio da internet usando o [Amazon Route 53](#) ou um registrador de domínios de terceiros da sua escolha. O nome de domínio personalizado pode ser o nome de um subdomínio ou do domínio raiz (também conhecido como “ápex da zona”) de um domínio da internet registrado.

O nome de domínio deve seguir a especificação [RFC 1035](#) e pode ter no máximo 63 octetos por etiqueta e 255 octetos no total.

### Certificados para nomes de domínio personalizados

Antes de configurar um nome de domínio personalizado para uma API, você deve ter um certificado SSL/TLS pronto no ACM. Se o ACM não estiver disponível na região da AWS onde você está criando o nome de domínio personalizado, será necessário importar um certificado para o API Gateway nessa região.

Para importar um certificado SSL/TLS, você deve fornecer o corpo do certificado SSL/TLS formatado em PEM, sua chave privada e a cadeia de certificado para o nome de domínio personalizado.

Cada certificado armazenado no ACM é identificado por seu ARN. Com certificados emitidos pelo ACM, não é necessário se preocupar em expor detalhes de certificados confidenciais, como a chave privada. Para usar um certificado gerenciado pela AWS para um nome de domínio, basta fazer referência ao seu ARN.

Se o seu aplicativo usa a fixação de certificados, às vezes chamada de fixação SSL, para fixar um certificado do ACM, talvez o aplicativo não consiga se conectar ao seu domínio após a AWS renovar o certificado. Para ter mais informações, consulte [Problemas de fixação do certificado](#) no Guia do usuário do AWS Certificate Manager.

### Nomes de domínio personalizados curinga

Com nomes de domínio personalizados curinga, você pode suportar um número quase infinito de nomes de domínio sem exceder a [cota padrão](#). Por exemplo, você pode dar a cada um de seus clientes seu próprio nome de domínio `customername.api.example.com`.

Para criar um nome de domínio personalizado curinga, especifique um curinga (\*) como o primeiro subdomínio de um domínio personalizado que representa todos os subdomínios possíveis de um domínio raiz.

Por exemplo, o nome de domínio personalizado curinga \*.example.com resulta em subdomínios, como a.example.com, b.example.com e c.example.com, que são todos roteados para o mesmo domínio.

Os nomes de domínio personalizados curinga oferecem suporte a configurações distintas dos nomes de domínio personalizados padrão do API Gateway. Por exemplo, em uma única conta da AWS, é possível configurar \*.example.com e a.example.com para se comportarem de forma diferente.

Você pode usar as variáveis de contexto `$context.domainName` e `$context.domainPrefix` para determinar o nome de domínio que um cliente usou para chamar sua API. Para saber mais sobre variáveis de contexto, consulte [Variáveis para transformações de dados para o API Gateway](#).

Para criar um nome de domínio personalizado curinga, é necessário fornecer um certificado emitido pelo ACM que foi validado usando o DNS ou o método de validação por e-mail.

#### Note

Não é possível criar um nome de domínio personalizado curinga se uma conta da AWS diferente tiver criado um nome de domínio personalizado que esteja em conflito com o nome de domínio personalizado curinga. Por exemplo, se a conta A tiver criado a.example.com, a conta B não poderá criar o nome de domínio personalizado curinga \*.example.com. Se a conta A e a conta B compartilham um proprietário, entre em contato com a [Central de Suporte da AWS](#) para solicitar uma exceção.

## Próximas etapas para nomes de domínio personalizados

Para configurar um nome de domínio personalizado para uma API HTTP, use a documentação da seção API REST do Guia do desenvolvedor do API Gateway.

Primeiro, especifique o certificado para o nome de domínio personalizado. Para obter mais informações, consulte [the section called “Preparar certificados no AWS Certificate Manager”](#). Depois, crie um nome de domínio regional personalizado. Para obter mais informações, consulte [the section called “Configurar um nome de domínio regional personalizado”](#).

## Mapear estágios de API para um nome de domínio personalizado para APIs de WebSocket

Você usa mapeamentos de API para conectar estágios de API a um nome de domínio personalizado. Depois de criar um nome de domínio e configurar registros DNS, você usa mapeamentos de API para enviar tráfego para as suas APIs utilizando o seu nome de domínio personalizado.

Um mapeamento de API especifica uma API, um estágio e, opcionalmente, um caminho a usar para o mapeamento. Por exemplo, você pode mapear o estágio `production` de uma API para `wss://api.example.com/orders`.

Antes de criar um mapeamento de API, você deve ter uma API, um estágio e um nome de domínio personalizado. Para saber mais sobre como criar um nome de domínio personalizado, consulte [the section called “Configurar um nome de domínio regional personalizado”](#).

### Restrições

- Em um mapeamento de API, o nome de domínio personalizado e as APIs mapeadas devem estar na mesma conta da AWS.
- Os mapeamentos de API devem conter apenas letras, números e os caracteres a seguir: `$-_.+!*'()`.
- O comprimento máximo para o caminho em um mapeamento de API é de 300 caracteres.
- Você não pode mapear APIs WebSocket para o mesmo nome de domínio personalizado que uma API HTTP ou API REST.
- Se você criar mapeamentos de API com vários níveis, o API Gateway mudará todos os nomes de cabeçalho para minúsculas.

### Crie um mapeamento de API

Para criar um mapeamento de API, você deve primeiro criar um nome de domínio personalizado, uma API e um estágio. Para obter informações sobre como criar um nome de domínio personalizado, consulte [the section called “Configurar um nome de domínio regional personalizado”](#).

### Console de gerenciamento da AWS

Para criar um mapeamento de API

1. Inicie uma sessão no console do API Gateway em <https://console.aws.amazon.com/apigateway>.

2. Escolha Custom domain names (Nomes de domínios personalizados).
3. Selecione um nome de domínio personalizado que você já criou.
4. Escolha API mappings (Mapeamentos de API).
5. Escolha Configure API mappings (Configurar mapeamentos de API).
6. Escolha Add new mapping (Adicionar novo mapeamento).
7. Insira uma API, um Estágio e, opcionalmente, um Caminho.
8. Escolha Salvar.

## AWS CLI

O comando [create-api-mapping](#) a seguir cria um mapeamento de API. Neste exemplo, o API Gateway envia solicitações para `api.example.com/v1` para a API e o estágio especificados.

```
aws apigatewayv2 create-api-mapping \
 --domain-name api.example.com \
 --api-mapping-key v1 \
 --api-id a1b2c3d4 \
 --stage test
```

## CloudFormation

O exemplo de CloudFormation a seguir cria um mapeamento de API.

```
MyApiMapping:
 Type: 'AWS::ApiGatewayV2::ApiMapping'
 Properties:
 DomainName: api.example.com
 ApiMappingKey: 'v1'
 ApiId: !Ref MyApi
 Stage: !Ref MyStage
```

## Tipos de endereço IP para nomes de domínio personalizados para APIs de WebSocket

Ao criar um nome de domínio personalizado, você especifica o tipo de endereços IP que podem invocar seu domínio. É possível escolher IPv4 para resolver endereços IPv4 e invocar seu domínio ou escolher pilha dupla para permitir que endereços IPv4 e IPv6 invoquem seu domínio.

Recomendamos que você defina o tipo de endereço IP como pilha dupla para aliviar o esgotamento do espaço IP ou para seu procedimento de segurança. Para ter mais informações sobre os benefícios de um tipo de endereço IP de pilha dupla, consulte [IPv6 na AWS](#).

## Considerações sobre tipos de endereço IP

As considerações a seguir podem afetar o uso de tipos de endereço IP.

- O tipo de endereço IP padrão para nomes de domínio personalizados do API Gateway é IPv4.
- Seu nome de domínio personalizado não precisa ter o mesmo tipo de endereço IP para todas as APIs associadas a ele. Se você desabilitar seu endpoint de API padrão, isso poderá afetar como os chamadores podem invocar sua API.

## Alterar o tipo de endereço IP do nome de domínio personalizado

É possível alterar o tipo de endereço IP atualizando a configuração do endpoint do nome de domínio. É possível atualizar a configuração do endpoint usando o Console de gerenciamento da AWS, a AWS CLI, o CloudFormation ou um SDK da AWS.

## Console de gerenciamento da AWS

Como alterar o tipo de endereço IP de um nome de domínio personalizado

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha um nome de domínio personalizado público.
3. Escolha Configurações de endpoint.
4. Em Tipo de endereço IP, escolha IPv4 ou Pilha dupla.
5. Escolha Salvar.

## AWS CLI

O comando [update-domain-name](#) a seguir atualiza uma API para ter um tipo de endereço IP de pilha dupla:

```
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
```



A saída será exibida da seguinte forma:

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

## Desabilitar o endpoint padrão para APIs de WebSocket

Por padrão, os clientes podem invocar sua API usando o endpoint `execute-api` gerado pelo API Gateway para sua API. Para garantir que os clientes possam acessar sua API somente usando um nome de domínio personalizado, desabilite o endpoint `execute-api` padrão. Quando o endpoint padrão é desabilitado, ele afeta todos os estágios de uma API.

O procedimento a seguir mostra como desabilitar o endpoint padrão de uma API de WebSocket.

### Console de gerenciamento da AWS

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha uma API de WebSocket.
3. Escolha Configurações da API.
4. Em Detalhes da API, escolha Editar.
5. Em Endpoint padrão, selecione Inativo.
6. Escolha Salvar alterações.
7. No painel de navegação principal, selecione Rotas.

8. Selecione Implantar e reimplante a API ou crie uma etapa para que a alteração entre em vigor.

## AWS CLI

O comando [update-api](#) indicado abaixo desabilita o endpoint padrão para uma API de WebSocket.

```
aws apigatewayv2 update-api \
 --api-id abcdef123 \
 --disable-execute-api-endpoint
```

Depois de desabilitar o endpoint padrão, é necessário implantar sua API para que a alteração entre em vigor.

O comando da AWS CLI a seguir cria uma implantação.

```
aws apigatewayv2 create-deployment \
 --api-id abcdef123 \
 --stage-name dev
```

## Proteger as APIs de WebSocket no API Gateway

O API Gateway fornece várias maneiras de proteger sua API de determinadas ameaças, como usuários mal-intencionados ou picos de tráfego. É possível proteger a API usando estratégias, como gerar certificados SSL ou configurar destinos de controle de utilização. Para ter mais informações sobre como gerar certificados, consulte [the section called “Certificados do cliente”](#). O restante desta seção aborda a configuração de destinos de controle de utilização.

Você pode configurar o controle de utilização para as suas APIs para ajudar a protegê-las da sobrecarga de numerosas solicitações. Os controles de utilização são aplicados de acordo com o melhor esforço e devem ser considerados alvos, e não limites máximos garantidos de solicitações.

O API Gateway controla a utilização das solicitações para a sua API usando o algoritmo do bucket de token, em que um token equivale a uma solicitação. Especificamente, o API Gateway analisa a taxa e uma intermitência de envios de solicitações de todas as APIs na sua conta, por região. No algoritmo do bucket de token, uma intermitência pode permitir a saturação predefinida desses limites, mas há alguns casos em que outros fatores também podem exceder tais limites.

Quando os envios de solicitações excederem a taxa de solicitação de estado fixo e os limites de intermitência, o API Gateway iniciará o controle de utilização de solicitações. Neste momento, pode ser que os clientes recebam respostas de erro 429 Too Many Requests. Ao capturar essas exceções, o cliente poderá reenviar as solicitações com falha de uma forma que restrinja as taxas.

Como desenvolvedor de APIs, você pode definir os limites alvo para estágios ou rotas de APIs particulares para melhorar a performance geral em todas as APIs na sua conta.

## Controle de utilização no nível da conta por região

Por padrão, o API Gateway controla a utilização das solicitações de estado fixo por segundo (RPS) em todas as APIs de uma conta da AWS, por região. Ele também limita a intermitência (ou seja, o tamanho máximo do bucket) em todas as APIs de uma conta da AWS, por região. No API Gateway, o limite de intermitência representa o número máximo alvo de envios simultâneos de solicitações que ele fará antes de retornar respostas de erro 429 Too Many Requests. Para obter mais informações sobre cotas de controle de utilização, consulte [Cotas e observações importantes](#).

Os limites por conta são aplicados a todas as APIs em uma conta em uma região especificada. O limite de taxas no nível da conta pode ser aumentado por meio de uma solicitação; é possível obter limites mais altos com APIs com tempos limite mais curtos e cargas úteis menores. Para solicitar um aumento nos limites de controle de utilização no nível da conta por região, entre em contato com a [Central de Suporte da AWS](#). Para obter mais informações, consulte [Cotas e observações importantes](#). Observe que tais limites não podem ser superiores aos limites de controle de utilização da AWS.

## Limitação em nível de rota

Você pode definir a limitação no nível da rota para substituir as limitações de solicitação no nível da conta para um estágio específico ou para rotas individuais em sua API. Os limites de controle de utilização da rota padrão não podem exceder os limites de taxa em nível de conta.

É possível configurar o controle de utilização de nível de rota usando a AWS CLI. O comando [update-stage](#) indicado abaixo configura o controle de utilização personalizado para o estágio especificado e a rota de uma API:

```
aws apigatewayv2 update-stage \
 --api-id a1b2c3d4 \
 --stage-name dev \
 --route-settings '{"messages":
{"ThrottlingBurstLimit":100, "ThrottlingRateLimit":2000}}'
```

# Monitorar as APIs de WebSocket no API Gateway

É possível usar métricas do CloudWatch e o CloudWatch Logs para monitorar APIs WebSocket. Combinando logs e métricas, você pode registrar erros em log e monitorar o desempenho da API.

## Note

O API Gateway pode não gerar logs e métricas nos seguintes casos:

- Erros 413 de entidade de solicitação muito grande
- Erros 429 de muitas solicitações excessivos
- Erros da série 400 de solicitações enviadas a um domínio personalizado que não tem mapeamento de API
- Erros da série 500 causados por falhas internas

## Tópicos

- [Monitorar a execução de APIs de WebSocket com métricas do CloudWatch](#)
- [Configurar o registro em log para APIs de WebSocket no API Gateway](#)

## Monitorar a execução de APIs de WebSocket com métricas do CloudWatch

É possível usar as métricas do [Amazon CloudWatch](#) para monitorar as APIs WebSocket. A configuração é semelhante à usada para APIs REST. Para obter mais informações, consulte [Monitorar a execução da API REST com métricas do Amazon CloudWatch](#).

As métricas a seguir são compatíveis com APIs WebSocket:

| Métrica      | Descrição                                                               |
|--------------|-------------------------------------------------------------------------|
| ConnectCount | O número de mensagens enviadas à integração de rotas \$connect.         |
| MessageCount | O número de mensagens enviadas à API WebSocket do cliente e vice-versa. |

| Métrica            | Descrição                                                                                                                                                                                                                 |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IntegrationError   | O número de solicitações que retornam uma resposta 4XX/5XX da integração.                                                                                                                                                 |
| ClientError        | O número de solicitações que têm uma resposta 4XX retornada pelo API Gateway antes de a integração ser invocada.                                                                                                          |
| ExecutionError     | Erros que ocorreram durante a chamada da integração.                                                                                                                                                                      |
| IntegrationLatency | A diferença de hora entre o envio da solicitação para integração por parte do API Gateway e o recebimento da resposta da integração por parte do API Gateway. Supressão para retornos de chamada e integrações simuladas. |

É possível usar as dimensões na tabela a seguir para filtrar métricas do API Gateway.

| Dimensão           | Descrição                                                                                             |
|--------------------|-------------------------------------------------------------------------------------------------------|
| Apild              | Filtra as métricas do API Gateway para uma API com o ID de API especificado.                          |
| ID da API, estágio | Filtra métricas do API Gateway para um estágio de API com o ID da API especificada e o ID do estágio. |

| Dimensão                         | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Apilid, método, recurso, estágio | <p>Filtra métricas do API Gateway para um método de API com o ID da API especificada, ID do estágio, caminho do recurso e ID da rota.</p> <p>O API Gateway não enviará essas métricas a menos que você tenha habilitado explicitamente métricas detalhadas do CloudWatch. Isso pode ser feito ao chamar a ação <a href="#">UpdateStage</a> da API REST V2 do API Gateway para atualizar a propriedade <code>detailedMetricsEnabled</code> como <code>true</code>. Como alternativa, você pode chamar o comando <a href="#">update-stage</a> da AWS CLI para atualizar a propriedade <code>DetailedMetricsEnabled</code> para <code>true</code>. Permitir essas métricas incorrerá em cobranças adicionais na conta. Para obter informações de definição de preço, consulte <a href="#">Definição de preço do Amazon CloudWatch</a>.</p> |

## Configurar o registro em log para APIs de WebSocket no API Gateway

É possível habilitar o registro em log para gravar logs no CloudWatch Logs. Há dois tipos de registro de API em logs no CloudWatch: registro de execução e de acesso. No registro de execução, o API

Gateway gerencia o CloudWatch Logs. O processo inclui a criação de grupos de log e fluxos de log, além de relatórios aos fluxos de log sobre solicitações e respostas de qualquer autor da chamada.

Para melhorar seu procedimento de segurança, recomendamos que você use o registro em log de execução no nível ERROR ou INFO. Você pode precisar fazer isso para cumprir vários requisitos de conformidade. Para ter mais informações, consulte [Amazon API Gateway controls](#) no Guia do usuário do AWS Security Hub.

No registro de acessos, você, assim como um desenvolvedor de API, registra quem acessou sua API e como o autor da chamada acessou a API. Você pode criar seu próprio grupo de logs ou escolher um existente que possa ser gerenciado pelo API Gateway. Para especificar os detalhes de acesso, selecione variáveis `$context` (expressas em um formato de sua escolha) e selecione um grupo de logs como o destino.

Para obter instruções sobre como configurar o registro em log do CloudWatch, consulte [the section called “Configurar o registro em log da API do CloudWatch usando o console do API Gateway”](#).

Quando você especifica o Formato de registro, é possível escolher em quais variáveis de contexto se registrar. As seguintes variáveis são compatíveis.

| Parâmetro                                            | Descrição                                                     |
|------------------------------------------------------|---------------------------------------------------------------|
| <code>\$context.apiId</code>                         | O identificador que o API Gateway atribui à sua API.          |
| <code>\$context.authorize.error</code>               | A mensagem de erro de autorização.                            |
| <code>\$context.authorize.latency</code>             | A latência de autorização em ms.                              |
| <code>\$context.authorize.status</code>              | O código de status retornado de uma tentativa de autorização. |
| <code>\$context.authorizer.error</code>              | A mensagem de erro retornada de um autorizador.               |
| <code>\$context.authorizer.integrationLatency</code> | A latência do autorizador do Lambda em ms.                    |
| <code>\$context.authorizer.integrationStatus</code>  | O código de status retornado de um autorizador do Lambda.     |

| Parâmetro                                         | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$context.authorizer.latency</code>         | A latência de autorizador em ms.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <code>\$context.authorizer.requestId</code>       | O ID da solicitação do endpoint da AWS.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <code>\$context.authorizer.status</code>          | O código de status retornado de um autorizador.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <code>\$context.authorizer.principalId</code>     | A identificação do usuário principal associada ao token enviado pelo cliente e retornado de uma função do Lambda do autorizador do Lambda do API Gateway. (Anteriormente, um autorizador do Lambda era conhecido como um autorizador personalizado.)                                                                                                                                                                                                                                                                                                                                       |
| <code>\$context.authorizer.<i>property</i></code> | <p>O valor transformado em string do par de chave/valor especificado do mapa context retornado de uma função de autorizador do Lambda do API Gateway. Por exemplo, se o autorizador retornar o seguinte mapa context:</p> <pre>"context" : {<br/>    "key":<br/>    "value",<br/>    "numKey":<br/>    1,<br/>    "boolKey":<br/>    true<br/>}</pre> <p>chamar <code>\$context.authorizer.key</code> retornará a string "value", chamar <code>\$context.authorizer.numKey</code> retornará a string "1" e chamar <code>\$context.authorizer.boolKey</code> retornará a string "true".</p> |



| Parâmetro                                          | Descrição                                                                                                                             |
|----------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$context.authenticate.error</code>          | A mensagem de erro retornada de uma tentativa de autenticação.                                                                        |
| <code>\$context.authenticate.latency</code>        | A latência de autenticação em ms.                                                                                                     |
| <code>\$context.authenticate.status</code>         | O código de status retornado de uma tentativa de autenticação.                                                                        |
| <code>\$context.connectedAt</code>                 | O tempo de conexão formatado em <a href="#">Epoch</a> .                                                                               |
| <code>\$context.connectionId</code>                | Um ID exclusivo para a conexão que pode ser utilizado para fazer uma chamada ao cliente.                                              |
| <code>\$context.domainName</code>                  | Um nome de domínio para a API WebSocket . Pode ser usado para fazer um retorno de chamada ao cliente (em vez de um valor codificado). |
| <code>\$context.error.message</code>               | Uma string que contém uma mensagem de erro do API Gateway.                                                                            |
| <code>\$context.error.messageString</code>         | O valor citado de <code>\$context.error.message</code> , ou seja, " <code>\$context.error.message</code> " .                          |
| <code>\$context.error.responseType</code>          | O tipo de resposta de erro.                                                                                                           |
| <code>\$context.error.validationErrorString</code> | Uma string que contém uma mensagem de erro de validação detalhada.                                                                    |
| <code>\$context.eventType</code>                   | O tipo de evento: CONNECT, MESSAGE ou DISCONNECT .                                                                                    |
| <code>\$context.extendedRequestId</code>           | Equivale a <code>\$context.requestId</code> .                                                                                         |
| <code>\$context.identity.accountId</code>          | O ID da conta da AWS associado à solicitação.                                                                                         |

| Parâmetro                                                     | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$context.identity.apiKey</code>                        | A chave do proprietário da API associada à solicitação de API habilitada por chave.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <code>\$context.identity.apiKeyId</code>                      | O ID da chave da API associada à solicitação de API habilitada por chave                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <code>\$context.identity.caller</code>                        | O identificador da entidade do autor da chamada que assinou a solicitação. Compatível com rotas que usam a autorização do IAM.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <code>\$context.identity.cognitoAuthenticationProvider</code> | <p>Uma lista separada por vírgulas de todos os provedores de autenticação do Amazon Cognito usados pelo autor da chamada que faz a solicitação. Disponível somente se a solicitação foi assinada com credenciais do Amazon Cognito.</p> <p>Por exemplo, para uma identidade e de um grupo de usuários do Amazon Cognito, cognito-idp. <i>region</i>.amazonaws.com/ <i>user_pool_id</i> ,cognito-idp. <i>region</i>.amazonaws.com/ <i>user_pool_id</i> :CognitoSignIn: <i>token subject claim</i></p> <p>Consulte informações sobre os provedores de autenticação do Amazon Cognito disponíveis em <a href="#">Using Federated Identities</a> no Guia do desenvolvedor do Amazon Cognito.</p> |

| Parâmetro                                                 | Descrição                                                                                                                                                                                                                                                                                                                       |
|-----------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$context.identity.cognitoAuthenticationType</code> | O tipo de autenticação do Amazon Cognito do autor da chamada que faz a solicitação. Disponível somente se a solicitação foi assinada com credenciais do Amazon Cognito. Os valores possíveis incluem <code>authenticated</code> para identidades autenticadas e <code>unauthenticated</code> para identidades não autenticadas. |
| <code>\$context.identity.cognitoIdentityId</code>         | O ID de identidade do Amazon Cognito do autor da chamada que faz a solicitação. Disponível somente se a solicitação foi assinada com credenciais do Amazon Cognito.                                                                                                                                                             |
| <code>\$context.identity.cognitoIdentityPoolId</code>     | O ID do grupo de identidades do Amazon Cognito do autor da chamada que faz a solicitação. Disponível somente se a solicitação foi assinada com credenciais do Amazon Cognito.                                                                                                                                                   |
| <code>\$context.identity.principalOrgId</code>            | O <a href="#">ID da organização da AWS</a> . Compatível com rotas que usam a autorização do IAM.                                                                                                                                                                                                                                |
| <code>\$context.identity.sourceIp</code>                  | O endereço IP de origem da conexão TCP que está fazendo a solicitação para ao API Gateway.                                                                                                                                                                                                                                      |
| <code>\$context.identity.user</code>                      | O identificador da entidade do usuário que será autorizado contra o acesso a recursos. Compatível com rotas que usam a autorização do IAM.                                                                                                                                                                                      |
| <code>\$context.identity.userAgent</code>                 | O agente de usuário do autor da chamada da API.                                                                                                                                                                                                                                                                                 |

| Parâmetro                                            | Descrição                                                                                                                                                                                                        |
|------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$context.identity.userArn</code>              | O Nome do Recurso Amazon (ARN) do usuário efetivo identificado após a autenticação.                                                                                                                              |
| <code>\$context.integration.error</code>             | A mensagem de erro retornada de uma integração.                                                                                                                                                                  |
| <code>\$context.integration.integrationStatus</code> | Para a integração de proxy do Lambda, o código de status retornado pelo AWS Lambda, e não pelo código de função do Lambda de backend.                                                                            |
| <code>\$context.integration.latency</code>           | A latência de integração em ms. Equivale a <code>\$context.integrationLatency</code> .                                                                                                                           |
| <code>\$context.integration.requestId</code>         | O ID da solicitação do endpoint da AWS. Equivale a <code>\$context.awsEndpointRequestId</code> .                                                                                                                 |
| <code>\$context.integration.status</code>            | O código de status retornado de uma integração. Para integrações de proxy do Lambda, esse é o código de status que seu código de função do Lambda retorna. Equivale a <code>\$context.integrationStatus</code> . |
| <code>\$context.integrationLatency</code>            | A latência de integração em ms, disponível somente para registro de log de acesso.                                                                                                                               |
| <code>\$context.messageId</code>                     | Um ID exclusivo do servidor para uma mensagem. Disponível apenas quando o <code>\$context.eventType</code> é MESSAGE.                                                                                            |
| <code>\$context.requestId</code>                     | Igual a <code>\$context.extendedRequestId</code> .                                                                                                                                                               |
| <code>\$context.requestTime</code>                   | O horário da solicitação <a href="#">CLF</a> formatado (dd/MM/yyyy:HH:mm:ss +-hhmm).                                                                                                                             |

| Parâmetro                               | Descrição                                                                     |
|-----------------------------------------|-------------------------------------------------------------------------------|
| <code>\$context.requestTimeEpoch</code> | O tempo de solicitação formatado em <a href="#">Epoch</a> , em milissegundos. |
| <code>\$context.routeKey</code>         | A chave de roteamento selecionada.                                            |
| <code>\$context.stage</code>            | O estágio de implantação da chamada da API (por exemplo, beta ou prod).       |
| <code>\$context.status</code>           | O status da resposta.                                                         |
| <code>\$context.waf.error</code>        | A mensagem de erro retornada pelo AWS WAF.                                    |
| <code>\$context.waf.latency</code>      | A latência do AWS WAF em ms.                                                  |
| <code>\$context.waf.status</code>       | O código de status retornado pelo AWS WAF.                                    |

Exemplos de alguns formatos de log de acesso comumente usados são mostrados no console do API Gateway e estão listados a seguir.

- CLF ([Formato de log comum](#)):

```
$context.identity.sourceIp $context.identity.caller \
$context.identity.user [$context.requestTime] "$context.eventType $context.routeKey
$context.connectionId" \
$context.status $context.requestId
```

Os caracteres de continuação (\) têm o objetivo de ajuda visual. O formato do log deve ser uma única linha. É possível adicionar um caractere de nova linha (\n) no final do formato de log para incluir uma nova linha no final de cada entrada de log.

- JSON:

```
{
 "requestId": "$context.requestId", \
 "ip": "$context.identity.sourceIp", \
 "caller": "$context.identity.caller", \
 "user": "$context.identity.user", \
}
```

```
"requestTime":"$context.requestTime", \
"eventType":"$context.eventType", \
"routeKey":"$context.routeKey", \
"status":"$context.status", \
"connectionId":"$context.connectionId"
}
```

Os caracteres de continuação (\) têm o objetivo de ajuda visual. O formato do log deve ser uma única linha. É possível adicionar um caractere de nova linha (\n) no final do formato de log para incluir uma nova linha no final de cada entrada de log.

- XML:

```
<request id="$context.requestId"> \
 <ip>$context.identity.sourceIp</ip> \
 <caller>$context.identity.caller</caller> \
 <user>$context.identity.user</user> \
 <requestTime>$context.requestTime</requestTime> \
 <eventType>$context.eventType</eventType> \
 <routeKey>$context.routeKey</routeKey> \
 <status>$context.status</status> \
 <connectionId>$context.connectionId</connectionId> \
</request>
```

Os caracteres de continuação (\) têm o objetivo de ajuda visual. O formato do log deve ser uma única linha. É possível adicionar um caractere de nova linha (\n) no final do formato de log para incluir uma nova linha no final de cada entrada de log.

- CSV (valores separados por vírgula):

```
$context.identity.sourceIp,$context.identity.caller, \
$context.identity.user,$context.requestTime,$context.eventType, \
$context.routeKey,$context.connectionId,$context.status, \
$context.requestId
```

Os caracteres de continuação (\) têm o objetivo de ajuda visual. O formato do log deve ser uma única linha. É possível adicionar um caractere de nova linha (\n) no final do formato de log para incluir uma nova linha no final de cada entrada de log.

# Referência de nome de recurso da Amazon (ARN) do API Gateway

As tabelas a seguir listam os nomes de recursos da Amazon (ARNs) para recursos do API Gateway. Para saber mais sobre como usar ARNs em políticas do AWS Identity and Access Management, consulte [Como o Amazon API Gateway funciona com o IAM](#) e [Controlar o acesso a uma API REST com permissões do IAM](#).

## Recursos de API HTTP e API WebSocket

| Recurso           | ARN                                                                                                                               |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| AccessLogSettings | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/apis/ <i>api-id</i> /<br>stages/ <i>stage-name</i> /accesslo<br>gsettings |
| API               | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/apis/ <i>api-id</i>                                                       |
| APIs              | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/apis                                                                      |
| Autorizador       | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/apis/ <i>api-id</i> /authoriz<br>ers/ <i>id</i>                           |
| Authorizers       | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/apis/ <i>api-id</i> /authoriz<br>ers                                      |
| Cors              | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/apis/ <i>api-id</i> /cors                                                 |

| Recurso              | ARN                                                                                                                            |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------|
| Implantação          | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/apis/ <i>api-id</i> /deployments/ <i>id</i>                            |
| Implantações         | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/apis/ <i>api-id</i> /deployments                                       |
| ExportedAPI          | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/apis/ <i>api-id</i> /exports/ <i>specification</i>                     |
| Integração           | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/apis/ <i>api-id</i> /integrations/ <i>integration-id</i>               |
| Integrações          | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/apis/ <i>api-id</i> /integrations                                      |
| IntegrationResponse  | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/apis/ <i>api-id</i> /integrationresponses/ <i>integration-response</i> |
| IntegrationResponses | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/apis/ <i>api-id</i> /integrationresponses                              |
| Modelo               | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/apis/ <i>api-id</i> /models/ <i>id</i>                                 |
| Modelos              | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/apis/ <i>api-id</i> /models                                            |



| Recurso               | ARN                                                                                                                                     |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| ModelTemplate         | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/apis/ <i>api-id</i> /models/ <i>id</i> /template                                |
| Rota                  | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/apis/ <i>api-id</i> /routes/ <i>id</i>                                          |
| Rotas                 | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/apis/ <i>api-id</i> /routes                                                     |
| RouteRequestParameter | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/apis/ <i>api-id</i> /routes/ <i>id</i> /requestparameters/ <i>key</i>           |
| RouteResponse         | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/apis/ <i>api-id</i> /routes/ <i>id</i> /routeresponses/ <i>id</i>               |
| RouteResponses        | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/apis/ <i>api-id</i> /routes/ <i>id</i> /routeresponses                          |
| RouteSettings         | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/apis/ <i>api-id</i> /stages/ <i>stage-name</i> /routesettings/ <i>route-key</i> |
| Estágio               | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/apis/ <i>api-id</i> /stages/ <i>stage-name</i>                                  |
| Estágios              | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/apis/ <i>api-id</i> /stages                                                     |
| VpcLink               | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/vpclinks/ <i>vpclink-id</i>                                                     |

| Recurso  | ARN                                                              |
|----------|------------------------------------------------------------------|
| VpcLinks | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/vpclinks |

## Recursos de API REST

| Recurso            | ARN                                                                                                         |
|--------------------|-------------------------------------------------------------------------------------------------------------|
| Conta              | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/account                                             |
| ApiKey             | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/apikeys/ <i>id</i>                                  |
| ApiKeys            | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/apikeys                                             |
| Autorizador        | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/restapis/ <i>api-id</i> /<br>authorizers/ <i>id</i> |
| Authorizers        | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/restapis/ <i>api-id</i> /<br>authorizers            |
| ClientCertificate  | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/clientcertifica<br>tes/ <i>id</i>                   |
| ClientCertificates | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/clientcertificates                                  |
| Implantação        | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/restapis/ <i>api-id</i> /<br>deployments/ <i>id</i> |

| Recurso               | ARN                                                                                                                                                             |
|-----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Implantações          | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/restapis/ <i>api-id</i> /<br>deployments                                                                |
| DocumentationPart     | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/restapis/ <i>api-id</i> /<br>documentation/parts/ <i>id</i>                                             |
| DocumentationParts    | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/restapis/ <i>api-id</i> /<br>documentation/parts                                                        |
| DocumentationVersion  | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/restapis/ <i>api-id</i> /<br>documentation/versions/ <i>version</i>                                     |
| DocumentationVersions | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/restapis/ <i>api-id</i> /<br>documentation/versions                                                     |
| GatewayResponse       | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/restapis/ <i>api-id</i> /<br>gatewayresponses/ <i>response-type</i>                                     |
| GatewayResponses      | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/restapis/ <i>api-id</i> /<br>gatewayresponses                                                           |
| Integração            | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/restapis/ <i>api-id</i> /<br>resources/ <i>resource-id</i> /methods/<br><i>http-method</i> /integration |

| Recurso             | ARN                                                                                                                                                                                               |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IntegrationResponse | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/restapis/ <i>api-id</i> /<br>resources/ <i>resource-id</i> /methods/<br><i>http-method</i> /integration/respo<br>nses/ <i>status-code</i> |
| Método              | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/restapis/ <i>api-id</i> /<br>resources/ <i>resource-id</i> /methods/<br><i>http-method</i>                                                |
| MethodResponse      | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/restapis/ <i>api-id</i> /<br>resources/ <i>resource-id</i> /methods/<br><i>http-method</i> /responses/ <i>status-co</i><br><i>de</i>      |
| Modelo              | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/restapis/ <i>api-id</i> /<br>models/ <i>model-name</i>                                                                                    |
| Modelos             | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/restapis/ <i>api-id</i> /<br>models                                                                                                       |
| RequestValidator    | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/restapis/ <i>api-id</i> /<br>requestvalidators/ <i>id</i>                                                                                 |
| RequestValidators   | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/restapis/ <i>api-id</i> /<br>requestvalidators                                                                                            |
| Recurso             | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/restapis/ <i>api-id</i> /<br>resources/ <i>id</i>                                                                                         |

| Recurso      | ARN                                                                                                            |
|--------------|----------------------------------------------------------------------------------------------------------------|
| Recursos     | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/restapis/ <i>api-id</i> /<br>resources                 |
| RestApi      | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/restapis/ <i>api-id</i>                                |
| RestApis     | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/restapis                                               |
| Estágio      | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/restapis/ <i>api-id</i> /<br>stages/ <i>stage-name</i> |
| Estágios     | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/restapis/ <i>api-id</i> /<br>stages                    |
| Tags         | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/tags/ <i>url-encoded-resource-arn</i>                  |
| Modelo       | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/restapis/models<br>/ <i>model-name</i> /template       |
| UsagePlan    | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/usageplans/ <i>usageplan-id</i>                        |
| UsagePlans   | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/usageplans                                             |
| UsagePlanKey | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/usageplans/ <i>usageplan-id</i> /keys/ <i>id</i>       |

| Recurso       | ARN                                                                                           |
|---------------|-----------------------------------------------------------------------------------------------|
| UsagePlanKeys | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/usageplans/ <i>usageplan-id</i> /keys |
| VpcLink       | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/vpclinks/ <i>vpclink-id</i>           |
| VpcLinks      | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/vpclinks                              |

## Recursos de nomes de domínio

| Recurso          | ARN                                                                                                                        |
|------------------|----------------------------------------------------------------------------------------------------------------------------|
| ApiMapping       | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/domainnames/ <i>domain-name</i> /apimappings/ <i>id</i>            |
| ApiMappings      | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/domainnames/ <i>domain-name</i> /apimappings                       |
| BasePathMapping  | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/domainnames/ <i>domain-name</i> /basepathmappings/ <i>basepath</i> |
| BasePathMappings | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/domainnames/ <i>domain-name</i> /basepathmappings                  |
| DomainName       | arn: <i>partition</i> :apigatew<br>ay: <i>region</i> ::/domainnames/ <i>domain-name</i>                                    |

| Recurso                      | ARN                                                                                                                                                                   |
|------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DomainNameAccessAssociation  | arn: <i>partition</i> :apigateway: <i>region</i> : <i>account-id</i> :/domainnameaccessassociations/domainname/ <i>domain-name</i> /vpcesource/ <i>vpce-source-id</i> |
| DomainNameAccessAssociations | arn: <i>partition</i> :apigateway: <i>region</i> : <i>account-id</i> :/domainnameaccessassociations                                                                   |
| DomainNames                  | arn: <i>partition</i> :apigateway: <i>region</i> ::/domainnames                                                                                                       |
| RoutingRule                  | arn: <i>partition</i> :apigateway: <i>region</i> : <i>account-id</i> :/domainnames/ <i>domain-name</i> /routingrules/ <i>routing-rule-id</i>                          |

## execute-api (APIs HTTP, WebSocket e REST)

| Recurso                          | ARN                                                                                                                                              |
|----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| Endpoint de API WebSocket        | arn: <i>partition</i> :execute-api: <i>region</i> : <i>account-id</i> : <i>api-id</i> / <i>stage</i> / <i>route-key</i>                          |
| Endpoint de API REST e API HTTP* | arn: <i>partition</i> :execute-api: <i>region</i> : <i>account-id</i> : <i>api-id</i> / <i>stage</i> / <i>http-method</i> / <i>resource-path</i> |
| Autorizador do Lambda**          | arn: <i>partition</i> :execute-api: <i>region</i> : <i>account-id</i> : <i>api-id</i> /authorizers/ <i>authorizer-id</i>                         |

\* O ARN do endpoint da rota `$default` para APIs HTTP é `arn:partition:execute-api:region:account-id:api-id/*/$default`.

\* Esse ARN é aplicável somente ao definir a condição `SourceArn` na [política de recursos](#) de uma função de autorizador do Lambda. Para ver um exemplo, consulte [the section called “Criar um autorizador do Lambda”](#).



# Extensões OpenAPI para o API Gateway

As extensões do API Gateway são compatíveis com a autorização específica da AWS e as integrações de API específicas do API Gateway para APIs REST e APIs HTTP. Nesta seção, descreveremos as extensões do API Gateway para a especificação do OpenAPI.

## Tip

Para entender como as extensões do API Gateway são usadas em uma aplicação, você pode usar o console do API Gateway para criar uma API REST ou HTTP e exportá-la para um arquivo de definição do OpenAPI. Para obter mais informações sobre como exportar uma API, consulte [Exportar uma API REST do API Gateway](#) e [Exportar APIs HTTP do API Gateway](#).

## Tópicos

- [Objeto x-amazon-apigateway-any-method](#)
- [Objeto x-amazon-apigateway-cors](#)
- [Propriedade x-amazon-apigateway-api-key-source](#)
- [Objeto x-amazon-apigateway-auth](#)
- [Objeto x-amazon-apigateway-authorizer](#)
- [Propriedade x-amazon-apigateway-authtype](#)
- [Propriedade 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](#)
- [Propriedade x-amazon-apigateway-request-validator](#)
- [Objeto x-amazon-apigateway-request-validators](#)
- [Objeto x-amazon-apigateway-request-validators.requestValidator](#)
- [x-amazon-apigateway-security-policy](#)
- [Propriedade x-amazon-apigateway-tag-value](#)

## Objeto x-amazon-apigateway-any-method

Especifica o [Objeto de operação do OpenAPI](#) para o método ANY genérico do API Gateway em um [Objeto de item de caminho do OpenAPI](#). Esse objeto pode existir junto com outros objetos de operação e obterá qualquer método HTTP que não tenha sido explicitamente declarado.

A tabela a seguir lista as propriedades estendidas pelo API Gateway. Para as outras propriedades de Operação do OpenAPI, consulte a especificação OpenAPI.

| Nome da propriedade         | Tipo    | Descrição                                                                                                                                                                          |
|-----------------------------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>isDefaultRoute</code> | Boolean | Especifica se uma rota é a rota do <code>\$default</code> . Compatível apenas com APIs HTTP. Para saber mais, consulte <a href="#">Criar rotas para APIs HTTP no API Gateway</a> . |

| Nome da propriedade             | Tipo                                                   | Descrição                                                                                                                                                                                                    |
|---------------------------------|--------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x-amazon-apigateway-integration | <a href="#">Objeto x-amazon-apigateway-integration</a> | Especifica a integração do método com o backend. Esta é uma propriedade estendida do objeto de <a href="#">Operação do OpenAPI</a> . A integração pode ser de tipo AWS, AWS_PROXY, HTTP, HTTP_PROXY ou MOCK. |

## Exemplos de x-amazon-apigateway-any-method

O exemplo a seguir integra o método ANY em um recurso de proxy, {proxy+}, com uma função do 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"
 }
 }
}
```

O exemplo a seguir cria uma rota \$default para uma API HTTP que se integra a uma função do 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 a configuração de compartilhamento de recursos entre origens (CORS) para uma API HTTP. A extensão é aplicável à estrutura OpenAPI em nível de raiz. Para saber mais, consulte [Configurar CORS para APIs HTTP no API Gateway](#).

| Nome da propriedade | Tipo    | Descrição                                                                                                          |
|---------------------|---------|--------------------------------------------------------------------------------------------------------------------|
| allowOrigins        | Array   | Especifica as origens permitidas.                                                                                  |
| allowCredentials    | Boolean | Especifica se as credenciais estão incluídas na solicitação de CORS.                                               |
| exposeHeaders       | Array   | Especifica os cabeçalhos que estão expostos.                                                                       |
| maxAge              | Integer | Especifica o número de segundos que o navegador deve armazenar em cache os resultados da solicitação de simulação. |

| Nome da propriedade | Tipo  | Descrição                              |
|---------------------|-------|----------------------------------------|
| allowMethods        | Array | Especifica os métodos HTTP permitidos. |
| allowHeaders        | Array | Especifica os cabeçalhos permitidos.   |

## Exemplo de x-amazon-apigateway-cors

Veja a seguir um exemplo de configuração de CORS para uma 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"
]
}
```

## Propriedade x-amazon-apigateway-api-key-source

Especifique a origem para receber uma chave de API a fim de controlar os métodos de API que exigem uma chave. Essa propriedade no nível da API é do tipo String. Para saber mais sobre

como configurar um método para exigir uma chave de API, consulte [the section called “Configurar um método para usar chaves de API com uma definição OpenAPI”](#).

Especifique a origem da chave de API para as solicitações. Os valores válidos são:

- HEADER para receber a chave de API do cabeçalho X-API-Key de uma solicitação.
- AUTHORIZER para receber a chave da API do UsageIdentifierKey de um autorizador do Lambda (anteriormente conhecido como autorizador personalizado).

## Exemplo de x-amazon-apigateway-api-key-source

O exemplo a seguir define o cabeçalho X-API-Key como a origem da chave de 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 x-amazon-apigateway-auth

Define um tipo de autenticação a ser aplicado para a autenticação de invocações de método no API Gateway.

| Nome da propriedade | Tipo   | Descrição                                                                                                                                                                    |
|---------------------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| type                | string | Especifica o tipo de autorização. Especifique "NONE" para acesso aberto. Especifique "AWS_IAM" para usar permissões do IAM. Os valores diferenciam maiúsculas de minúsculas. |

## Exemplo de x-amazon-apigateway-auth

O exemplo a seguir define o tipo de autenticação para um 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 um autorizador do Lambda, um grupo de usuários do Amazon Cognito ou um autorizador do JWT a ser aplicado para a autorização de invocações de método no API Gateway. Esta extensão se aplica à definição de segurança no [OpenAPI 2](#) e ao esquema de segurança no [OpenAPI 3](#).

| Nome da propriedade | Tipo   | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------------------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| type                | string | <p>O tipo de autorizador. Essa é uma propriedade obrigatória.</p> <p>Para APIs REST, especifique token para um autorizador com a identidade do autor da chamada incorporada em um token de autorização. Especifique request para um autorizador com a identidade do autor da chamada contida nos parâmetros da solicitação. Especifique cognito_user_pools para um autorizador que use um grupo de usuários do Amazon Cognito para controlar o acesso à API.</p> <p>Para APIs HTTP, especifique request para um autorizador</p> |



| Nome da propriedade                | Tipo                | Descrição                                                                                                                                                                                                                                                     |
|------------------------------------|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                    |                     | do Lambda com a identidade do autor da chamada contida nos parâmetros da solicitação. Especifique <code>jwt</code> para um autorizador do JWT.                                                                                                                |
| <code>authorizerUri</code>         | <code>string</code> | <p>O URI (Uniform Resource Identifier) da função do Lambda do autorizador. A sintaxe é a seguinte:</p> <pre>"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"</pre> |
| <code>authorizerCredentials</code> | <code>string</code> | <p>As credenciais necessárias para invocar o autorizador, se houver, no formato de um ARN de uma função de execução do IAM. Por exemplo, <code>"arn:aws:iam::account-id:IAM_role"</code>.</p>                                                                 |

| Nome da propriedade                         | Tipo                 | Descrição                                                                                                                                                                                                                                                                                                                                                                                                         |
|---------------------------------------------|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>authorizerPayloadFormatVersion</code> | <code>string</code>  | Para APIs HTTP, especifique o formato dos dados que o API Gateway envia para um autorizador do Lambda e como o API Gateway interpreta a resposta do Lambda. Para saber mais, consulte <a href="#">the section called “Versão do formato da carga útil”</a> .                                                                                                                                                      |
| <code>enableSimpleResponses</code>          | <code>Boolean</code> | Para APIs HTTP, especifica se um autorizador <code>request</code> retorna um valor booleano ou uma política do IAM. Compatível apenas com autorizadores com um <code>authorizerPayloadFormatVersion</code> de 2.0. Se habilitado, a função do autorizador do Lambda retornará um valor booleano. Para saber mais, consulte <a href="#">the section called “Resposta de função do Lambda para o formato 2.0”</a> . |
| <code>identitySource</code>                 | <code>string</code>  | Lista de expressões de mapeamento separadas por vírgulas dos parâmetros de solicitação como a origem de identidade. Aplicável somente para o autorizador dos tipos <code>"request"</code> e <code>jwt</code> .                                                                                                                                                                                                    |

| Nome da propriedade          | Tipo                 | Descrição                                                                                                                                                                                                |
|------------------------------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| jwtConfiguration             | Object               | Especifica o emissor e os públicos para um autorizador de JWT. Para saber mais, consulte <a href="#">JWTConfiguration</a> na Referência de API do API Gateway versão 2. Compatível apenas com APIs HTTP. |
| identityValidationExpression | string               | Uma expressão regular para validar o token como identidade e de entrada. Por exemplo, "^x-[a-z]+". Compatível somente com autorizadores TOKEN para APIs REST.                                            |
| authorizerResultTtlInSeconds | string               | O número de segundos durante os quais o resultado do autorizador é armazenado em cache.                                                                                                                  |
| providerARNs                 | Uma matriz de string | Uma lista dos ARNs do grupo de usuários do Amazon Cognito para COGNITO_USER_POOLS.                                                                                                                       |

## Exemplos de x-amazon-apigateway-authorizer para APIs REST

O exemplo de definições de segurança do OpenAPI a seguir especifica um autorizador do Lambda do tipo “token” chamado `test-authorizer`.

```
"securityDefinitions" : {
 "test-authorizer" : {
 "type" : "apiKey",
 "apiKey" for an API Gateway API. // Required and the value must be
```

```

 "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
 }
}
}

```

O trecho de objeto de operação do OpenAPI a seguir define o GET /http para usar o autorizador do Lambda anterior.

```

"/http" : {
 "get" : {
 "responses" : { },
 "security" : [{
 "test-authorizer" : []
 }],
 "x-amazon-apigateway-integration" : {
 "type" : "http",
 "responses" : {
 "default" : {
 "statusCode" : "200"
 }
 },
 "httpMethod" : "GET",
 "uri" : "http://api.example.com"
 }
 }
}
}

```

O exemplo de definições de segurança do OpenAPI a seguir especifica um autorizador do Lambda do tipo “solicitação”, com um único parâmetro de cabeçalho (auth) como origem de identidade. O `securityDefinitions` é chamado `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
 }
 }
}
```

As definições de segurança demonstrativas do OpenAPI a seguir especificam um autorizador do Lambda do tipo “solicitação”, com um cabeçalho (HeaderAuth1) e um parâmetro de string de consulta QueryString1 como origens de identidade.

```
"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
 }
}
}

```

As definições de segurança demonstrativas do OpenAPI a seguir especificam um autorizador do Lambda do API Gateway do tipo “solicitação”, com uma única variável de estágio (stage) como origem de identidade.

```

"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
 }
 }
}

```

O exemplo a seguir de definição de segurança da OpenAPI especifica um grupo de usuários do 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"
]
 }
 }
}

```

O trecho de objeto de operação da OpenAPI a seguir define o GET /http para usar o grupo de usuários anterior do Amazon Cognito como autorizador, sem escopos personalizados.

```

"/http" : {
 "get" : {
 "responses" : { },
 "security" : [{
 "cognito-pool" : []
 }],
 "x-amazon-apigateway-integration" : {
 "type" : "http",
 "responses" : {
 "default" : {
 "statusCode" : "200"
 }
 },
 "httpMethod" : "GET",
 "uri" : "http://api.example.com"
 }
 }
}

```

## Exemplos de x-amazon-apigateway-authorizer para APIs HTTP

O exemplo de OpenAPI 3.0 a seguir cria um autorizador JWT para uma API HTTP que usa o Amazon Cognito como um provedor de identidade, com o cabeçalho Authorization como uma origem de identidade.

```

"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"
 }
 }
}

```

O exemplo de OpenAPI 3.0 a seguir produz o mesmo autorizador de JWT como o exemplo anterior. No entanto, esse exemplo usa a propriedade `openIdConnectUrl` do OpenAPI para detectar o emissor automaticamente. O `openIdConnectUrl` deve estar totalmente formado.

```

"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"
 }
 }
}

```

O exemplo a seguir cria um autorizador do Lambda para uma API HTTP. Este exemplo de autorizador usa o cabeçalho `Authorization` como sua fonte de identidade. O autorizador usa a



versão do formato de carga 2.0 e retorna o valor booleano, porque `enableSimpleResponses` está definido como `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
 }
 }
}
```

## Propriedade x-amazon-apigateway-authtype

Para APIs REST, essa extensão pode ser usada para definir um tipo personalizado de um autorizador do Lambda. Nesse caso, o valor é de formato livre. Por exemplo, uma API pode ter vários autorizadores do Lambda que usam esquemas internos diferentes. Você pode usar essa extensão para identificar o esquema interno de um autorizador do Lambda.

Mais comumente, em APIs HTTP e APIs REST, ele também pode ser usado como uma maneira de definir a autorização do IAM em várias operações que compartilham o mesmo esquema de segurança. Neste caso, o termo `awsSigv4` é um termo reservado, juntamente com qualquer termo prefixado por `aws`.

Esta extensão se aplica ao esquema `apiKey` de segurança de tipo no [OpenAPI 2](#) e [OpenAPI 3](#).

## Exemplo de x-amazon-apigateway-authtype

O exemplo do OpenAPI 3 a seguir define a autorização do IAM em vários recursos em uma API REST ou 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"
 }
 }
}
```

O exemplo do OpenAPI 3 a seguir define um autorizador do Lambda com um esquema personalizado para uma 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"
}
```

## Consulte também

[authorizer.authType](#)

## Propriedade x-amazon-apigateway-binary-media-types

Especifica a lista de tipos de mídia binários que serão compatíveis com o API Gateway, como `application/octet-stream` e `image/jpeg`. Essa extensão é uma matriz JSON. Ela deve ser incluída como uma extensão de fornecedor de nível superior ao documento OpenAPI.

### Exemplo de x-amazon-apigateway-binary-media-types

O exemplo a seguir mostra a ordem de pesquisa de codificação de uma API.

```
"x-amazon-apigateway-binary-media-types": ["application/octet", "image/jpeg"]
```

## Objeto x-amazon-apigateway-documentation

Define as partes de documentação a serem importadas para o API Gateway. Esse objeto é um objeto JSON que contém uma matriz das instâncias de `DocumentationPart`.

| Nome da propriedade             | Tipo   | Descrição                                                                             |
|---------------------------------|--------|---------------------------------------------------------------------------------------|
| <code>documentationParts</code> | Array  | Uma matriz das instâncias de <code>DocumentationPart</code> exportadas ou importadas. |
| <code>version</code>            | String | O identificador de versão do snapshot das partes de documentação exportadas.          |

### Exemplo de x-amazon-apigateway-documentation

O exemplo a seguir da extensão do API Gateway para o OpenAPI define instâncias `DocumentationParts` a serem importadas ou exportadas de uma API no 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 o modo de acesso ao endpoint para uma API REST ou nome de domínio personalizado. Para saber mais sobre o acesso ao endpoint, consulte [Modo de acesso ao endpoint](#).

## x-amazon-apigateway-endpoint-access-modeExemplo de

O exemplo a seguir especifica o modo de acesso ao endpoint como estrito.

```
"x-amazon-apigateway-endpoint-access-mode": "STRICT"
```

## Objeto x-amazon-apigateway-endpoint-configuration

Especifica detalhes da configuração do endpoint de uma API. Esta extensão é uma propriedade estendida do objeto de [Operação do OpenAPI](#). Esse objeto deve estar presente em [extensões de fornecedor de nível superior](#) para o Swagger 2.0. Para a OpenAPI 3.0, ele deve estar presente nas extensões de fornecedor do [objeto Server](#).

| Nome da propriedade                    | Tipo                              | Descrição                                                                                                                                                                                                                                                                                                                                                        |
|----------------------------------------|-----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>disableExecuteApiEndpoint</code> | Booleano                          | Especifica se os clientes podem invocar sua API usando o endpoint <code>execute-api</code> padrão. Por padrão, os clientes podem invocar sua API com o endpoint <code>https://{api_id}.execute-api.{region}.amazonaws.com</code> padrão. Para exigir que os clientes usem um nome de domínio personalizado para invocar sua API, especifique <code>true</code> . |
| <code>vpcEndpointIds</code>            | Uma matriz de <code>String</code> | Uma lista de identificadores de <code>VpcEndpoint</code> nos quais criar aliases do Route 53 para uma API REST. Ele só é compatível com APIs REST do tipo de endpoint <code>PRIVATE</code> .                                                                                                                                                                     |
| <code>ipAddressType</code>             | <code>string</code>               | Os tipos de endereço IP que podem invocar uma API HTTP. Use <code>ipv4</code> para permitir que tipos de endereço IPv4 invoquem uma API HTTP. Use <code>dualstack</code> para permitir que tipos de endereço IPv4 e IPv6 invoquem uma API HTTP. É compatível apenas com APIs HTTP.                                                                               |

## Exemplos de x-amazon-apigateway-endpoint-configuration

O exemplo a seguir associa VPC endpoints especificados à API REST.

```
"x-amazon-apigateway-endpoint-configuration": {
 "vpcEndpointIds": ["vpce-0212a4ababd5b8c3e", "vpce-01d622316a7df47f9"]
}
```

O exemplo a seguir desabilita o endpoint padrão para uma API.

```
"x-amazon-apigateway-endpoint-configuration": {
 "disableExecuteApiEndpoint": true
}
```

O exemplo a seguir define o tipo de endereço IP como pilha dupla para uma API HTTP.

```
"x-amazon-apigateway-endpoint-configuration": {
 "ipAddressType": "dualstack"
}
```

## Objeto x-amazon-apigateway-gateway-responses

Define as respostas de gateway para uma API como um mapa de string para [GatewayResponse](#) de pares de chave/valor. A extensão é aplicável à estrutura OpenAPI em nível de raiz.

| Nome da propriedade | Tipo                                                                  | Descrição                                                   |
|---------------------|-----------------------------------------------------------------------|-------------------------------------------------------------|
| <i>responseType</i> | <a href="#">x-amazon-apigateway-gateway-responses.gatewayResponse</a> | Um GatewayResponse para o <i>responseType</i> especificado. |

## Exemplo de x-amazon-apigateway-gateway-responses

A extensão demonstrativa do API Gateway para OpenAPI a seguir define um mapa [GatewayResponses](#) que contém duas instâncias de [GatewayResponse](#), uma para o tipo DEFAULT\_4XX e outra para o 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 uma resposta de gateway de um tipo de resposta especificado, incluindo o código de status, todos os parâmetros de resposta aplicáveis ou modelos de resposta.

| Nome da propriedade       | Tipo                                                                     | Descrição                                                                                                                                                                                                                                     |
|---------------------------|--------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>responseParameters</i> | <a href="#">x-amazon-apigateway-gateway-responses.responseParameters</a> | Especifica os parâmetros <a href="#">GatewayResponse</a> , ou seja, os parâmetros de cabeçalho. Os valores de parâmetros podem usar qualquer valor de <a href="#">parâmetro de solicitação</a> de entrada ou um valor personalizado estático. |
| <i>responseTemplates</i>  | <a href="#">x-amazon-apigateway-gateway-responses.responseTemplates</a>  | Especifica os modelos de mapeamento da resposta de gateway. Os modelos não são                                                                                                                                                                |



| Nome da propriedade | Tipo   | Descrição                                        |
|---------------------|--------|--------------------------------------------------|
|                     |        | processados pelo mecanismo VTL.                  |
| <i>statusCode</i>   | string | Um código de status HTTP da resposta do gateway. |

## Exemplo de x-amazon-apigateway-gateway-responses.gatewayResponse

O exemplo a seguir da extensão do API Gateway para OpenAPI define uma [GatewayResponse](#) para personalizar a resposta `INVALID_API_KEY` de forma a retornar o código de status de 456, o valor do cabeçalho `api-key` da solicitação de entrada e a mensagem "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 um mapa de string para string de pares de chave/valor para gerar parâmetros de resposta de gateway a partir dos parâmetros da solicitação de entrada ou usando strings literais. Compatível apenas com APIs REST.

| Nome da propriedade                                        | Tipo   | Descrição                                                               |
|------------------------------------------------------------|--------|-------------------------------------------------------------------------|
| gatewayresponse. <i>param-position</i> . <i>param-name</i> | string | <i>param-position</i> pode ser header, path ou querystring . Para obter |

| Nome da propriedade | Tipo | Descrição                                                                                           |
|---------------------|------|-----------------------------------------------------------------------------------------------------|
|                     |      | mais informações, consulte <a href="#">Mapeamento de parâmetros para APIs REST no API Gateway</a> . |

## x-amazon-apigateway-gateway-responses.responseParameters example

O exemplo de extensões do OpenAPI a seguir mostra uma expressão de mapeamento de parâmetros de resposta [GatewayResponse](#) para habilitar o suporte a CORS para recursos nos domínios `*.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 modelos de mapeamento [GatewayResponse](#), como um mapa de string para string de pares de chave/valor para uma determinada resposta de gateway. Para cada par chave-valor, a chave é o tipo de conteúdo. Por exemplo, “application/json” e o valor são um modelo de mapeamento stringified para substituições de variáveis simples. Um modelo de mapeamento GatewayResponse não é processado pelo mecanismo [Velocity Template Language \(VTL\)](#).

| Nome da propriedade | Tipo   | Descrição                                                                                                            |
|---------------------|--------|----------------------------------------------------------------------------------------------------------------------|
| <i>content-type</i> | string | Um modelo de mapeamento de corpo GatewayResponse que oferece suporte apenas à substituição de variáveis simples para |

| Nome da propriedade | Tipo | Descrição                                     |
|---------------------|------|-----------------------------------------------|
|                     |      | personalizar um corpo de resposta de gateway. |

## Exemplo de x-amazon-apigateway-gateway-responses.responseTemplates

O exemplo de extensões do OpenAPI a seguir mostra um modelo de mapeamento

[GatewayResponse](#) para personalizar uma resposta de erro gerada pelo API Gateway em um formato específico do aplicativo.

```
"responseTemplates": {
 "application/json": "{ \"message\": $context.error.messageString, \"type\": $context.error.responseType, \"statusCode\": '488' }"
}
```

O exemplo de extensões do OpenAPI a seguir mostra um modelo de mapeamento

[GatewayResponse](#) para substituir uma resposta de erro gerada pelo API Gateway por uma mensagem de erro estática.

```
"responseTemplates": {
 "application/json": "{ \"message\": 'API-specific errors' }"
}
```

## x-amazon-apigateway-importexport-version

Especifica a versão do algoritmo de importação e exportação do API Gateway para APIs HTTP.

Atualmente, o único valor compatível é 1.0. Para saber mais, consulte [exportVersion](#) na Referência de API do API Gateway Versão 2.

## x-amazon-apigateway-importexport-version example

O exemplo a seguir define a versão de importação e exportação como 1.0.

```
{
 "openapi": "3.0.1",
 "x-amazon-apigateway-importexport-version": "1.0",
```

```
"info": { ...
```

## Objeto x-amazon-apigateway-integration

Especifica detalhes da integração de backend usada para esse método. Esta extensão é uma propriedade estendida do objeto de [Operação do OpenAPI](#). O resultado é um objeto de [integração do API Gateway](#).

| Nome da propriedade | Tipo                 | Descrição                                                                                                                                                                                                                            |
|---------------------|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| cacheKeyParameters  | Uma matriz de string | Uma lista de parâmetros de solicitação cujos valores devem ser armazenados em cache.                                                                                                                                                 |
| cacheNamespace      | string               | Um grupo de tags específico de API dos parâmetros em cache relacionados.                                                                                                                                                             |
| connectionId        | string               | O ID de um <a href="#">VpcLink</a> para a integração privada.                                                                                                                                                                        |
| connectionType      | string               | O tipo de conexão da integração. O valor válido é "VPC_LINK" para integração privada ou "INTERNET" , para outras.                                                                                                                    |
| credentials         | string               | Para credenciais baseadas em função IAM da AWS, especifique o ARN de uma função IAM apropriada. Se não forem especificadas, as credenciais assumirão como padrão permissões baseadas em recursos que devem ser adicionadas manualmen |

| Nome da propriedade | Tipo   | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|---------------------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     |        | <p>te para permitir que a API acesse o recurso. Para obter mais informações, consulte <a href="#">Conceder permissões com o uso de uma política de recursos</a>.</p> <p>Observação: ao usar credenciais do IAM, verifique se os <a href="#">endpoints regionais STS da AWS</a> estão habilitados para a região na qual essa API está implantada para obter a melhor performance.</p>                                                                                                                             |
| contentHandling     | string | <p>Tipos de conversão de codificação da carga de solicitação. Os valores válidos são 1) <code>CONVERT_TEXT</code> , para converter uma carga binária em uma string codificada em base64 ou converter uma carga de texto em uma string codificada por <code>utf-8</code> ou transferir a carga de texto de forma nativa sem modificação e 2) <code>CONVERT_BINARY</code> para converter uma carga de texto em um blob decodificado em base64 ou transferir uma carga binária de forma nativa sem modificação.</p> |

| Nome da propriedade | Tipo   | Descrição                                                                                                                                                                                                                               |
|---------------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| httpMethod          | string | O método HTTP usado na solicitação de integração. Para invocações de função do Lambda, o valor deve ser POST.                                                                                                                           |
| integrationSubtype  | string | Especifica o subtipo para uma integração de serviço da AWS. Compatível apenas com APIs HTTP. Para obter subtipos de integração compatíveis, consulte <a href="#">the section called “AWSReferência de integrações de serviço da ”</a> . |
| integrationTarget   | string | O receptor do ALB ou do NLB para o qual enviar a solicitação. Aceito apenas em integrações privadas que usem links da VPC V2. Para obter mais informações, consulte <a href="#">the section called “Links de VPC V2”</a> .              |

| Nome da propriedade  | Tipo   | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|----------------------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| passthroughBehavior  | string | Especifica como uma carga de solicitação de um tipo de conteúdo não mapeado é transmitida na solicitação de integração sem modificação. Os valores com suporte <code>when_no_templates</code> , <code>when_no_match</code> e <code>never</code> . Para obter mais informações, consulte <a href="#">Integration.passthroughBehavior</a> .                                                                                                                              |
| payloadFormatVersion | string | Especifica o formato da carga enviada para uma integração. Obrigatório para APIs HTTP. Para APIs HTTP, os valores suportados para integrações de proxy do Lambda são <code>1.0</code> e <code>2.0</code> . Para todas as outras integrações, <code>1.0</code> é o único valor suportado. Para saber mais, consulte <a href="#">the section called “AWS LambdaIntegrações do ”</a> e <a href="#">the section called “AWSReferência de integrações de serviço da ”</a> . |

| Nome da propriedade            | Tipo                                                                     | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------------------------------|--------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>requestParameters</code> | <a href="#">Objeto x-amazon-apigateway-integration.requestParameters</a> | <p>Para APIs REST, especifica mapeamentos de parâmetros de solicitação de método para parâmetros de solicitação de integração. Parâmetros de solicitação com suporte são <code>queryString</code>, <code>path</code>, <code>header</code> e <code>body</code>.</p> <p>Para APIs HTTP, parâmetros de solicitação são um mapa de chave-valor que especifica parâmetros que são passados para integrações <code>AWS_PROXY</code> com um <code>integrationSubtype</code> especificado. Você pode fornecer valores estáticos ou mapear dados de solicitação, variáveis de estágio ou variáveis de contexto que são avaliadas no tempo de execução. Para saber mais, consulte <a href="#">the section called “AWSIntegrações de serviços da ”</a>.</p> |
| <code>requestTemplates</code>  | <a href="#">Objeto x-amazon-apigateway-integration.requestTemplates</a>  | Modelos de mapeamento para uma carga de solicitação de tipos MIME especificados.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |



| Nome da propriedade  | Tipo                                                             | Descrição                                                                                                                                                                                                                                                                                                                                                                                                        |
|----------------------|------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| responses            | <a href="#">Objeto x-amazon-apigateway-integration.responses</a> | Define as respostas do método e especifica mapeamentos de parâmetros desejados ou mapeamentos de carga de respostas de integração para respostas de método.                                                                                                                                                                                                                                                      |
| responseTransferMode | string                                                           | O modo de transferência de resposta da integração. Utilize BUFFERED para o API Gateway esperar receber a resposta completa antes de iniciar a transmissão. Use STREAM para o API Gateway enviar respostas parciais de volta ao cliente assim que elas se tornem disponíveis. Para saber mais, consulte <a href="#">the section called “Transmitir a resposta de integração para suas integrações de proxy”</a> . |
| timeoutInMillis      | integer                                                          | Tempos limite para a integração entre 50 ms e 29.000 ms.                                                                                                                                                                                                                                                                                                                                                         |

| Nome da propriedade | Tipo                                                                            | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|---------------------|---------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| type                | string                                                                          | <p>O tipo de integração com o backend especificado. Os valores válidos são:</p> <ul style="list-style-type: none"><li>• http ou http_proxy , para integração com um backend HTTP.</li><li>• aws_proxy , para integração com funções do AWS Lambda.</li><li>• aws, para integração com funções do AWS Lambda ou outros serviços da AWS, como o Amazon DynamoDB, Amazon Simple Notification Service ou Amazon Simple Queue Service.</li><li>• mock, para integração com o API Gateway sem invocar nenhum backend.</li></ul> <p>Para obter mais informações sobre os tipos de integração, consulte <a href="#">integration:type</a>.</p> |
| tlsConfig           | <a href="#">the section called “x-amazon-apigateway-integration.tls Config”</a> | Especifica a configuração de TLS para uma integração.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

| Nome da propriedade | Tipo   | Descrição                                                                                                                                                                       |
|---------------------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| uri                 | string | O URI de endpoint do backend. Para integrações do tipo aws, este é um valor de ARN. Para a integração o HTTP, esta é a URL do endpoint HTTP, incluindo o esquema https ou http. |

## Exemplos de x-amazon-apigateway-integration

Para APIs HTTP, é possível definir integrações na seção de componentes da sua definição do OpenAPI. Para saber mais, consulte [Objeto x-amazon-apigateway-integrations](#).

```
"x-amazon-apigateway-integration": {
 "$ref": "#/components/x-amazon-apigateway-integrations/integration1"
}
```

O exemplo a seguir cria uma integração com uma função do Lambda. Para fins de demonstração, supõe-se que os modelos de mapeamento de amostra mostrados em requestTemplates e responseTemplates dos exemplos abaixo apliquem a seguinte carga em formato JSON: { "name": "value\_1", "key": "value\_2", "redirect": { "url" : "..."} } para gerar uma saída JSON de { "stage": "value\_1", "user-id": "value\_2" } ou uma saída XML de <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> "
```



## Objeto x-amazon-apigateway-integrations

Define uma coleção de integrações. Você pode definir integrações na seção de componentes da definição do OpenAPI e reutilizar as integrações para várias rotas. Compatível apenas com APIs HTTP.

| Nome da propriedade  | Tipo                                                   | Descrição                             |
|----------------------|--------------------------------------------------------|---------------------------------------|
| <i>Integração do</i> | <a href="#">Objeto x-amazon-apigateway-integration</a> | Uma coleção de objetos de integração. |

## Exemplo de x-amazon-apigateway-integrations

O exemplo a seguir cria uma API HTTP que define duas integrações e faz referência às integrações usando `$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 modelos de mapeamento para uma carga de solicitação dos tipos MIME especificados.

| Nome da propriedade | Tipo   | Descrição                                                                                                                                                                                                         |
|---------------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>MIME type</i>    | string | Um exemplo do tipo MIME é <code>application/json</code> . Para obter informações sobre como criar um modelo de mapeamento, consulte <a href="#">the section called “Transformações de modelo de mapeamento”</a> . |

## Exemplo de x-amazon-apigateway-integration.requestTemplates

O exemplo a seguir define modelos de mapeamento para uma carga de solicitação dos tipos MIME `application/json` e `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 APIs REST, especifica mapeamentos de parâmetros de solicitação de método nomeados para parâmetros de solicitação de integração. Os parâmetros de solicitação do método devem ser definidos antes de serem referenciados.

Para APIs HTTP, especifica parâmetros que são transmitidos para integrações AWS\_PROXY com um `integrationSubtype` especificado.

| Nome da propriedade                                                                                  | Tipo   | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|------------------------------------------------------------------------------------------------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>integration.requestParameters</code><br><i>&lt;param-type&gt;</i><br><i>&lt;param-name&gt;</i> | string | Para APIs REST, normalmente, o valor é um parâmetro de solicitação de método predefinido do formato <code>method.request.&lt;param-type&gt;.&lt;param-name&gt;</code> em que <code>&lt;param-type&gt;</code> pode ser <code>queryString</code> , <code>path</code> , <code>header</code> ou <code>body</code> . No entanto <code>\$context.VARIABLE_NAME</code> , <code>\$stageVariables.VARIABLE_NAME</code> e <code>STATIC_VALUE</code> também são válidos. Para o parâmetro <code>body</code> , <code>&lt;param-name&gt;</code> é uma expressão de caminho JSON sem o prefixo <code>\$</code> . |
| <i>parameter</i>                                                                                     | string | Para APIs HTTP, parâmetros de solicitação são um mapa de chave-valor que                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |



| Nome da propriedade | Tipo | Descrição                                                                                                                                                                                                                                                                                                                                                                         |
|---------------------|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     |      | especifica parâmetros que são passados para integrações AWS_PROXY com um <code>integrationSubtype</code> especificado. Você pode fornecer valores estáticos ou mapear dados de solicitação, variáveis de estágio ou variáveis de contexto que são avaliadas no tempo de execução. Para saber mais, consulte <a href="#">the section called “AWSIntegrações de serviços da ”</a> . |

## x-amazon-apigateway-integration.requestParametersExemplo de

O seguinte exemplo de mapeamentos de parâmetros de solicitação converte os parâmetros de consulta (`version`), cabeçalho (`x-user-id`) e caminho (`service`) de uma solicitação de método nos parâmetros de consulta (`stage`), cabeçalho (`x-userid`) e caminho (`op`) de uma solicitação de integração, respectivamente.

### Note

Se você estiver criando recursos por meio do OpenAPI ou do CloudFormation, os valores estáticos deverão estar entre aspas simples.

Para adicionar esse valor no console, digite `application/json` na caixa, sem aspas.

```
"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 as respostas do método e especifica mapeamentos de parâmetros ou mapeamentos de carga de respostas de integração para respostas de método.

| Nome da propriedade                 | Tipo                                                            | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-------------------------------------|-----------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Padrão do status de resposta</i> | <a href="#">Objeto x-amazon-apigateway-integration.response</a> | <p>Uma expressão regular usada para corresponder a resposta de integração à resposta do método ou default para capturar qualquer resposta que você não tenha configurado. Para integrações HTTP, essa expressão regular é aplicada ao código de status da resposta de integração. Para invocações do Lambda, a expressão se aplica ao campo <code>errorMessage</code> do objeto de informações de erro retornado por AWS Lambda como um corpo de resposta de falha quando a execução da função do Lambda lança uma exceção.</p> <div> <b>Note</b><br/>O nome da propriedade de do <i>Padrão do status de</i></div> |



```
 "method.response.header.Location": "integration.response.body.redirect.url"
 }
}
```

## Objeto x-amazon-apigateway-integration.response

Define uma resposta e especifica mapeamentos de parâmetros ou mapeamentos de carga a partir da resposta de integração para a resposta de método.

| Nome da propriedade | Tipo                                                                      | Descrição                                                                                                                                                                          |
|---------------------|---------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| statusCode          | string                                                                    | Código de status HTTP para a resposta de método; por exemplo, "200". Isso deve equivaler a uma resposta correspondente no campo responses da <a href="#">Operação do OpenAPI</a> . |
| responseTemplates   | <a href="#">Objeto x-amazon-apigateway-integration.responseTemplates</a>  | Especifica modelos de mapeamento específicos de tipo MIME para a carga da resposta.                                                                                                |
| responseParameters  | <a href="#">Objeto x-amazon-apigateway-integration.responseParameters</a> | Especifica mapeamentos de parâmetros para a resposta. Somente os parâmetros header e body da resposta de integração podem ser mapeados para os parâmetros header do método.        |
| contentHandling     | string                                                                    | Tipos de conversão de codificação da carga de                                                                                                                                      |

| Nome da propriedade | Tipo | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|---------------------|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     |      | resposta. Os valores válidos são 1) <code>CONVERT_T0_TEXT</code> , para converter uma carga binária em uma string codificada em base64 ou converter uma carga de texto em uma string codificada por <code>utf-8</code> ou transferir a carga de texto de forma nativa sem modificação e 2) <code>CONVERT_T0_BINARY</code> para converter uma carga de texto em um blob decodificado em base64 ou transferir uma carga binária de forma nativa sem modificação. |

## x-amazon-apigateway-integration.responseExemplo de

O exemplo a seguir define uma resposta 302 para o método que deriva uma carga do tipo MIME do `application/json` ou do `application/xml` no backend. A resposta usa os modelos de mapeamento fornecidos e retorna a URL de redirecionamento da resposta de integração no cabeçalho `Location` do 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 modelos de mapeamento para uma carga de resposta dos tipos MIME especificados.

| Nome da propriedade | Tipo   | Descrição                                                                                                                                                                                                                                                                                                                                                       |
|---------------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>MIME type</i>    | string | Especifica um modelo de mapeamento para transformar o corpo da resposta de integração no corpo da resposta de método para um determinado tipo MIME. Para obter informações sobre como criar um modelo de mapeamento, consulte <a href="#">the section called “Transformações de modelo de mapeamento”</a> . Um exemplo do <i>tipo MIME</i> é application/json . |

## Exemplo de x-amazon-apigateway-integration.responseTemplate

O exemplo a seguir define modelos de mapeamento para uma carga de solicitação dos tipos MIME application/json e 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 mapeamentos de parâmetros de resposta de método de integração para parâmetros de resposta de método. Você pode mapear header, body ou valores estáticos no tipo header da resposta do método. Compatível apenas com APIs REST.

| Nome da propriedade                               | Tipo   | Descrição                                                                                                        |
|---------------------------------------------------|--------|------------------------------------------------------------------------------------------------------------------|
| method.response.header. <i>&lt;param-name&gt;</i> | string | O valor do parâmetro nomeado pode ser derivado dos tipos header e body dos parâmetros de resposta de integração. |

### x-amazon-apigateway-integration.responseParametersExemplo de

O exemplo a seguir mapeia parâmetros body e header da resposta de integração para dois parâmetros header da resposta de 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 a configuração de TLS para uma integração.

| Nome da propriedade      | Tipo    | Descrição                                                                |
|--------------------------|---------|--------------------------------------------------------------------------|
| insecureSkipVerification | Boolean | Compatível apenas com APIs REST. Especifica se o API Gateway ignorará ou |

| Nome da propriedade | Tipo | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------------------|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     |      | <p>não a verificação de que o certificado para um endpoint de integração é emitido por uma <a href="#">autoridade de certificação compatível</a>. Isso não é recomendado, mas permite que você use certificados assinados por autoridades de certificação privadas ou certificados autoassinados. Se habilitado, o API Gateway ainda executará a validação básica do certificado, que inclui verificar a data de expiração do certificado, o nome do host e a presença de uma autoridade de certificação raiz. O certificado raiz pertencente à autoridade privada deve atender às seguintes restrições:</p> <ul style="list-style-type: none"><li>• A extensão <code>x509 keyUsage</code> deve ter <code>keyCertSign</code> .</li><li>• A extensão <code>x509 basicConstraints</code> deve ter <code>CA:TRUE</code>.</li></ul> <p>Compatível apenas com integrações HTTP e HTTP_PROXY .</p> |



| Nome da propriedade             | Tipo                | Descrição                                                                                                                                                                                                                                                                                                                                                    |
|---------------------------------|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                 |                     | <p> <b>Warning</b></p> <p>Habilitar <code>insecureSkipVerification</code> não é recomendado, especialmente para integrações com endpoints HTTPS públicos. Se você habilitar <code>insecureSkipVerification</code>, você aumenta o risco de ataques man-in-the-middle.</p> |
| <code>serverNameToVerify</code> | <code>string</code> | Compatível apenas com integrações privadas de API HTTP. Se você especificar um nome de servidor, o API Gateway o usará para verificar o nome do host no certificado de integração. O nome do servidor também está incluído no handshake de TLS para oferecer suporte à indicação de nome de servidor (SNI) ou à hospedagem virtual.                          |

## Exemplos `x-amazon-apigateway-integration.tlsConfig`

O OpenAPI 3.0 demonstrativo a seguir permite `insecureSkipVerification` para uma integração 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",
}
```

O OpenAPI 3.0 demonstrativo a seguir especifica um `serverNameToVerify` para uma integração 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 o tamanho mínimo de compactação para um API REST. Para habilitar a compactação, especifique um inteiro entre 0 e 10485760. Para saber mais, consulte [Compressão de carga útil para APIs REST no API Gateway](#).

### Exemplo de x-amazon-apigateway-minimum-compression-size

O exemplo a seguir especifica um tamanho mínimo de compactação de 5242880 bytes para uma API REST.

```
"x-amazon-apigateway-minimum-compression-size": 5242880
```

## x-amazon-apigateway-policy

Especifica uma política de recursos para uma API REST. Para saber mais sobre as políticas de recursos, consulte [Controlar o acesso a uma API REST com políticas de recursos do API Gateway](#). Para obter políticas de recursos demonstrativas, consulte [Exemplos de política de recursos do API Gateway](#).

### x-amazon-apigateway-policyExemplo de

O exemplo a seguir especifica uma política de recursos para uma API REST. A política de recursos nega (bloqueia) o tráfego de entrada para uma API de um bloco de endereços IP de origem especificado. Na importação, "execute-api:/\*" é convertido para `arn:aws:execute-api:region:account-id:api-id/*`, usando a Região atual, o ID de sua conta da AWS e o ID da API REST atual.

```
"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"
 }
 }
 }
]
}
```

```

 }
]
}
```

## Propriedade x-amazon-apigateway-request-validator

Especifica um validador de solicitação, fazendo referência a um *request\_validator\_name* do mapa [Objeto x-amazon-apigateway-request-validators](#), para habilitar a validação de solicitações na API receptora ou em um método. O valor dessa extensão é uma string JSON.

Essa extensão pode ser especificada no nível de API ou no nível de método. O validador em nível de API aplica-se a todos os métodos, a menos que ela seja substituído pelo validador em nível de método.

### x-amazon-apigateway-request-validatorExemplo de

O exemplo a seguir aplica o validador de solicitação basic em nível de API e, ao mesmo tempo, aplica o validador de solicitação parameter-only na solicitação 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 os validadores de solicitação com suporte para a API receptora como um mapa entre um nome de validador e as regras de validação de solicitações associadas. Esta extensão se aplica a uma API REST.

| Nome da propriedade           | Tipo                                                                           | Descrição                                                                                                                                                                                                                                                                                                                                                                                           |
|-------------------------------|--------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>request_validator_name</i> | <a href="#">Objeto x-amazon-apigateway-request-validators.requestValidator</a> | <p>Especifica as regras de validação que consistem no validador nomeado. Por exemplo:</p> <pre> "basic" : {   "validate RequestBody" : true,   "validate RequestParameters" : true }, </pre> <p>Para aplicar esse validador a um método específico, faça referência ao nome do validador (basic) como o valor da propriedade <a href="#">Propriedade x-amazon-apigateway-request-validator</a>.</p> |

## x-amazon-apigateway-request-validatorsExemplo de

O exemplo a seguir mostra um conjunto de validadores de solicitação para uma API como um mapa entre um nome de validador e as regras de validação de solicitações associadas.

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 as regras de validação de um validador de solicitação como parte da definição do mapa [Objeto x-amazon-apigateway-request-validators](#).

| Nome da propriedade       | Tipo    | Descrição                                                                                         |
|---------------------------|---------|---------------------------------------------------------------------------------------------------|
| validateRequestBody       | Boolean | Especifica se o corpo da solicitação deve ser validado (true) ou não (false).                     |
| validateRequestParameters | Boolean | Especifica se os parâmetros de solicitação necessários devem ser validados (true) ou não (false). |

## x-amazon-apigateway-request-validators.requestValidatorExemplo de

O exemplo a seguir mostra um validador de solicitação somente para parâmetros:

```
"params-only": {
 "validateRequestBody" : false,
```

```
"validateRequestParameters" : true
}
```

## x-amazon-apigateway-security-policy

Especifica uma política de segurança para uma API REST. Se você criar uma política de segurança que comece com "SecurityPolicy\_", também deverá definir o [modo de acesso ao endpoint](#). Para saber mais sobre políticas de segurança, consulte [Políticas de segurança para APIs REST no API Gateway](#).

### x-amazon-apigateway-security-policyExemplo de

O exemplo a seguir especifica SecurityPolicy\_TLS13\_1\_3\_2025\_0 para uma API REST.

```
"x-amazon-apigateway-security-policy": "SecurityPolicy_TLS13_1_3_2025_09"
```

## Propriedade x-amazon-apigateway-tag-value

Especifica o valor de uma [tag da AWS](#) para uma API HTTP. É possível usar a propriedade x-amazon-apigateway-tag-value como parte do [objeto da tag OpenAPI](#) para especificar as tags da AWS para uma API HTTP. Se você especificar um nome de tag sem a propriedade x-amazon-apigateway-tag-value, o API Gateway criará uma tag com uma string vazia para um valor.

Para saber mais sobre marcação, consulte [Marcar recursos do API Gateway](#).

| Nome da propriedade           | Tipo   | Descrição                  |
|-------------------------------|--------|----------------------------|
| name                          | String | Especifica a chave da tag. |
| x-amazon-apigateway-tag-value | String | Especifica o valor da tag. |

### x-amazon-apigateway-tag-valueExemplo de

O exemplo a seguir especifica duas tags para uma API HTTP:

- "Proprietário": "Admin"

- "Prod": ""

```
"tags": [
 {
 "name": "Owner",
 "x-amazon-apigateway-tag-value": "Admin"
 },
 {
 "name": "Prod"
 }
]
```



# Segurança no Amazon API Gateway

A segurança da nuvem na AWS é a nossa maior prioridade. Como cliente da AWS, você contará com um data center e uma arquitetura de rede criados para atender aos requisitos das organizações com as maiores exigências de segurança.

A segurança é uma responsabilidade compartilhada entre a AWS e você. O [Modelo de Responsabilidade Compartilhada](#) descreve isso como segurança da nuvem e segurança na nuvem:

- **Segurança da nuvem:** a AWS é responsável pela proteção da infraestrutura que executa serviços AWS na Nuvem AWS. A AWS também fornece serviços que podem ser usados com segurança. Auditores de terceiros testam e verificam regularmente a eficácia da nossa segurança como parte dos [Programas de conformidade da AWS](#). Para saber mais sobre os programas de conformidade que se aplicam ao Amazon API Gateway; consulte os [Serviços da AWS no escopo por programa de conformidade](#).
- **Segurança na nuvem:** sua responsabilidade é determinada pelo serviço da AWS que você usa. Você também é responsável por outros fatores, incluindo a confidencialidade de seus dados, os requisitos da empresa e as leis e regulamentos aplicáveis.

Esta documentação ajuda você a entender como aplicar o modelo de responsabilidade compartilhada ao usar o API Gateway. Os tópicos a seguir mostram como configurar o API Gateway para atender aos seus objetivos de segurança e de conformidade. Você também aprenderá a usar outros serviços da AWS que ajudam a monitorar e proteger os recursos do API Gateway.

Para obter mais informações, consulte [Visão geral de segurança no Amazon API Gateway](#).

## Tópicos

- [Proteção de dados no Amazon API Gateway](#)
- [Gerenciamento de identidade e acesso para o Amazon API Gateway](#)
- [Registro em log e monitoramento no Amazon API Gateway](#)
- [Validação de conformidade do Amazon API Gateway](#)
- [Resiliência no Amazon API Gateway](#)
- [Segurança de infraestrutura no Amazon API Gateway](#)
- [Análise de vulnerabilidades no Amazon API Gateway](#)
- [Melhores práticas de segurança no Amazon API Gateway](#)

# Proteção de dados no Amazon API Gateway

O [modelo de responsabilidade compartilhada](#) da AWS se aplica à proteção de dados no Amazon API Gateway. Conforme descrito nesse modelo, a AWS é responsável por proteger a infraestrutura global que executa toda a Nuvem AWS. Você é responsável por manter o controle sobre seu conteúdo hospedado nessa infraestrutura. Você também é responsável pelas tarefas de configuração e gerenciamento de segurança dos Serviços da AWS que usa. Para obter mais informações sobre a privacidade de dados, consulte as [Perguntas Frequentes sobre Privacidade de Dados](#). Para obter mais informações sobre a proteção de dados na Europa, consulte a postagem do blog [AWS Shared Responsibility Model and GDPR](#) no Blog de segurança da AWS.

Para fins de proteção de dados, recomendamos que você proteja as credenciais da Conta da AWS e configure as contas de usuário individuais com Centro de Identidade do AWS IAM ou AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use uma autenticação multifator (MFA) com cada conta.
- Use SSL/TLS para se comunicar com os recursos da AWS. Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Configure a API e atividade do usuário logando com AWS CloudTrail. Para obter mais informações sobre como usar as trilhas do CloudTrail para capturar atividades da AWS, consulte [Working with CloudTrail trails](#) no Guia do Usuário do AWS CloudTrail.
- Use as soluções de criptografia AWS, juntamente com todos os controles de segurança padrão em Serviços da AWS.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados sigilosos armazenados no Amazon S3.
- Se você precisar de módulos criptográficos validados pelo FIPS 140-3 ao acessar a AWS por meio de uma interface de linha de comando ou uma API, use um endpoint do FIPS. Para obter mais informações sobre os endpoints FIPS disponíveis, consulte [Federal Information Processing Standard \(FIPS\) 140-3](#).

É altamente recomendável que nunca sejam colocadas informações de identificação confidenciais, como endereços de e-mail dos seus clientes, em marcações ou campos de formato livre, como um campo Nome. Isso inclui quando você trabalha com o API Gateway ou outros Serviços da AWS usando o console, a API, a AWS CLI ou os AWS SDKs. Quaisquer dados inseridos em tags ou

campos de texto de formato livre usados para nomes podem ser usados para logs de faturamento ou de diagnóstico. Se você fornecer um URL para um servidor externo, recomendamos fortemente que não sejam incluídas informações de credenciais no URL para validar a solicitação a esse servidor.

## Criptografia de dados no Amazon API Gateway

Proteção de dados refere-se à proteção de dados em trânsito (à medida que são transferidos para e do API Gateway) e em repouso (enquanto estão armazenados na AWS).

### Criptografia de dados em repouso no Amazon API Gateway

Se você optar por habilitar o armazenamento em cache para uma API REST, poderá habilitar a criptografia de cache. Para saber mais, consulte [Configurações de cache para APIs REST no API Gateway](#).

Consulte mais informações sobre proteção de dados na publicação do blog [AWS Shared Responsibility Model and GDPR](#) no Blog de segurança da AWS.

### Criptografia e decodificação da chave privada do seu certificado

Quando você cria um nome de domínio personalizado para APIs privadas, seu certificado do ACM e chave privada são criptografados usando uma chave do KMS gerenciada pela AWS que tem o alias `aws/acm`. Você pode ver o ID da chave com esse alias no console do AWS KMS, Chaves gerenciadas pela AWS.

O API Gateway não acessa diretamente seus recursos do ACM. Ele usa o AWS TLS Connection Manager para proteger e acessar as chaves privadas do seu certificado. Quando você usa seu certificado do ACM para criar um nome de domínio personalizado do API Gateway para APIs privadas, o API Gateway associa seu certificado ao AWS TLS Connection Manager. Isso é feito criando uma concessão no AWS KMS em relação à sua chave gerenciada pela AWS com o prefixo `aws/acm`. Uma concessão é um instrumento de política que permite que o TLS Connection Manager use chaves do KMS em operações de criptografia. A concessão permite que a entidade principal autorizada (TLS Connection Manager) chame as operações de concessão especificadas na chave do KMS para decifrar a chave privada do seu certificado. Em seguida, o TLS Connection Manager usa o certificado e a chave privada descriptografada (texto simples) para estabelecer uma conexão segura (sessão SSL/TLS) com clientes de serviços do API Gateway. Quando o certificado é desassociado de um nome de domínio personalizado do API Gateway para APIs privadas, a concessão é removida.

Se você quiser remover o acesso à chave do KMS, recomendamos substituir ou excluir o certificado do serviço usando o Console de gerenciamento da AWS ou o comando `update-service` na AWS CLI.

## Contexto de criptografia para o API Gateway

[Contexto de criptografia](#) é um conjunto opcional de pares de chave/valor que pode conter mais informações contextuais sobre o possível uso da chave privada. O AWS KMS vincula o contexto de criptografia aos dados criptografados e usa dados adicionais autenticados para permitir criptografia autenticada.

Quando suas chaves TLS são usadas com o API Gateway e o TLS Connection Manager, o nome do serviço API Gateway é incluído no contexto de criptografia usado para criptografar sua chave em repouso. A fim de verificar para qual nome de domínio personalizado do API Gateway seu certificado e sua chave privada estão sendo usados, visualize o contexto de criptografia em seus logs do CloudTrail, conforme apresentado na próxima seção, ou examine a guia Recursos associados no console do ACM.

Para descriptografar os dados, o mesmo contexto de criptografia é incluído na solicitação. O API Gateway usa o mesmo contexto de criptografia em todas as operações criptográficas do AWS KMS, no qual a chave é `aws:apigateway:arn` e o valor é o nome do recurso da Amazon (ARN) do recurso `PrivateDomainName` do API Gateway.

O seguinte exemplo mostra o contexto de criptografia na saída de uma operação 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"
```

## Criptografia de dados em trânsito no Amazon API Gateway

As APIs criadas com o Amazon API Gateway expõem apenas endpoints HTTPS. O API Gateway não é compatível com endpoints não criptografados (HTTP).

O API Gateway gerencia os certificados dos endpoints execute-api padrão. Ao configurar um nome de domínio personalizado, [você especifica o certificado para o nome de domínio](#). Como prática recomendada, não [fixe certificados](#).

Para obter maior segurança, é possível escolher uma versão mínima do protocolo TLS (Transport Layer Security) a ser aplicada ao seu domínio personalizado do API Gateway. As APIs WebSocket e HTTP são compatíveis apenas com TLS 1.2. Para saber mais, consulte [Escolher uma política de segurança para o domínio personalizado no API Gateway](#).

Também é possível configurar uma distribuição do Amazon CloudFront com um certificado SSL personalizado em sua conta e usá-la com APIs regionais. Depois, você pode configurar a política de segurança para a distribuição do CloudFront com TLS 1.1 ou superior com base nos seus requisitos de segurança e conformidade.

Para obter mais informações sobre proteção de dados, consulte [Proteger as APIs REST no API Gateway](#) e a publicação do blog [AWS Shared Responsibility Model and GDPR](#) no Blog de segurança da AWS.

## Privacidade do tráfego entre redes

Usando o Amazon API Gateway, é possível criar APIs REST privadas que podem ser acessadas somente a partir da Amazon Virtual Private Cloud (VPC). A VPC usa um [VPC endpoint de interface](#), que é uma interface de rede de endpoint criada na VPC. Com as [políticas de recursos](#), você pode permitir ou negar o acesso à sua API de alguns VPCs e VPC endpoints, incluindo nas contas da AWS. Cada endpoint pode ser usado para acessar várias APIs privadas. Você também pode usar o Direct Connect para estabelecer uma conexão a partir de uma rede local para a VPC da Amazon e acessar sua API privada nessa conexão. O tráfego para a API privada sempre usa conexões seguras e não deixa a rede da Amazon; ele é isolado da Internet pública. Para saber mais, consulte [the section called “APIs REST privadas”](#).

# Gerenciamento de identidade e acesso para o Amazon API Gateway

O AWS Identity and Access Management (IAM) é um serviço da AWS service (Serviço da AWS) que ajuda um administrador a controlar com segurança o acesso aos recursos da AWS. Os administradores do IAM controlam quem pode ser autenticado (conectado) e autorizado (ter permissões) a usar recursos do API Gateway. O IAM é um AWS service (Serviço da AWS) que pode ser usado sem custo adicional.

## Tópicos

- [Público](#)
- [Autenticação com identidades](#)
- [Gerenciar o acesso usando políticas](#)
- [Como o Amazon API Gateway funciona com o IAM](#)
- [Exemplos de políticas baseadas em identidade do Amazon API Gateway](#)
- [Exemplos de políticas baseadas em recursos do Amazon API Gateway](#)
- [Solução de problemas de identidade e acesso do Amazon API Gateway](#)
- [Usar funções vinculadas ao serviço para o API Gateway](#)

## Público

A forma como você usa o AWS Identity and Access Management (IAM) difere com base em seu perfil:

- Usuário do serviço: solicite permissões ao seu administrador se você não conseguir acessar os atributos (consulte [Solução de problemas de identidade e acesso do Amazon API Gateway](#)).
- Administrador do serviço: determine o acesso do usuário e envie solicitações de permissão (consulte [Como o Amazon API Gateway funciona com o IAM](#)).
- Administrador do IAM: escreva políticas para gerenciar o acesso (consulte [Exemplos de políticas baseadas em identidade do Amazon API Gateway](#)).

## Autenticação com identidades

A autenticação é a forma como fazer login na AWS usando suas credenciais de identidade. Você precisa se autenticar como o Usuário raiz da conta da AWS, como um usuário do IAM ou assumindo um perfil do IAM.

Você pode fazer login como uma identidade federada usando credenciais de uma fonte de identidade, como o Centro de Identidade do AWS IAM (Centro de Identidade do IAM), autenticação única ou credenciais do Google/Facebook. Para ter mais informações sobre como fazer login, consulte [Como fazer login em sua Conta da AWS](#) no Guia do usuário do Início de Sessão da AWS.

Para acesso programático, a AWS oferece um SDK e uma CLI para assinar solicitações criptograficamente. Para ter mais informações, consulte [AWS Signature Version 4 para solicitações de API](#) no Guia do usuário do IAM.

### Usuário-raiz Conta da AWS

Ao criar uma Conta da AWS, você começa com uma identidade de login única chamada usuário-raiz da Conta da AWS, que tem acesso total a todos os recursos e Serviços da AWS. É altamente recomendável não usar o usuário-raiz em tarefas diárias. Consulte as tarefas que exigem credenciais de usuário-raiz em [Tarefas que exigem credenciais de usuário-raiz](#) no Guia do usuário do IAM.

### Usuários e grupos do IAM

Um [usuário do IAM](#) é uma identidade com permissões específicas para uma única pessoa ou aplicação. É recomendável usar credenciais temporárias, em vez de usuários do IAM com credenciais de longo prazo. Para saber mais, consulte [Exigir que os usuários humanos usem a federação com um provedor de identidade para acessar a AWS usando credenciais temporárias](#) no Guia do usuário do IAM.

Um [grupo do IAM](#) especifica um conjunto de usuários do IAM e facilita o gerenciamento de permissões para grandes conjuntos de usuários. Para ter mais informações, consulte [Casos de uso de usuários do IAM](#) no Guia do usuário do IAM.

### Perfis do IAM

Uma [perfil do IAM](#) é uma identidade com permissões específicas que oferece credenciais temporárias. É possível assumir um perfil [alternando de um usuário para um perfil do IAM \(console\)](#) ou chamando uma operação de API AWS CLI ou AWS. Para saber mais, consulte [Métodos para assumir um perfil](#) no Manual do usuário do IAM.



Os perfis do IAM são úteis para acesso de usuário federado, permissões de usuário do IAM temporárias, acesso entre contas, acesso entre serviços e aplicações em execução no Amazon EC2. Consulte mais informações em [Acesso a recursos entre contas no IAM](#) no Guia do usuário do IAM.

## Gerenciar o acesso usando políticas

Você controla o acesso na AWS criando políticas e anexando-as a identidades ou recursos da AWS. Uma política define permissões quando associada a uma identidade ou a um recurso. A AWS avalia essas políticas quando a entidade principal faz uma solicitação. A maioria das políticas é armazenada na AWS como documentos JSON. Para ter mais informações sobre documentos de política JSON, consulte [Visão geral das políticas JSON](#) no Guia do usuário do IAM.

Por meio de políticas, os administradores especificam quem tem acesso a que, definindo qual entidade principal pode realizar ações em quais recursos e sob quais condições.

Por padrão, usuários e perfis não têm permissões. Um administrador do IAM cria políticas do IAM e as adiciona aos perfis, os quais os usuários podem então assumir. As políticas do IAM definem permissões, independentemente do método usado para realizar a operação.

### Políticas baseadas em identidade

As políticas baseadas em identidade são documentos de políticas de permissão JSON que você anexa a uma identidade (usuário, grupo ou perfil). Essas políticas controlam quais ações as identidades podem realizar, em quais recursos e sob quais condições. Para saber como criar uma política baseada em identidade, consulte [Definir permissões personalizadas do IAM com as políticas gerenciadas pelo cliente](#) no Guia do Usuário do IAM.

As políticas baseadas em identidade podem ser políticas em linha (incorporadas diretamente em uma única identidade) ou políticas gerenciadas (políticas autônomas anexadas a várias identidades). Para saber como escolher entre uma política gerenciada e políticas em linha, consulte [Escolher entre políticas gerenciadas e políticas em linha](#) no Guia do usuário do IAM.

### Políticas baseadas em recursos

Políticas baseadas em recursos são documentos de políticas JSON que você anexa a um recurso. Entre os exemplos estão políticas de confiança de perfil do IAM e políticas de bucket do Amazon S3. Em serviços compatíveis com políticas baseadas em recursos, os administradores de serviço podem usá-las para controlar o acesso a um recurso específico. É necessário [especificar uma entidade principal](#) em uma política baseada em recursos.



Políticas baseadas em recursos são políticas em linha localizadas nesse serviço. Não é possível usar as políticas gerenciadas pela AWS do IAM em uma política baseada em atributos.

## Listas de controle de acesso (ACLs)

As listas de controle de acesso (ACLs) controlam quais entidades principais (membros, usuários ou perfis da conta) têm permissões para acessar um recurso. As ACLs são semelhantes às políticas baseadas em recursos, embora não usem o formato de documento de política JSON.

Amazon S3, AWS WAF e Amazon VPC são exemplos de serviços que oferecem compatibilidade com ACLs. Para saber mais sobre ACLs, consulte [Visão geral da lista de controle de acesso \(ACL\)](#) no Guia do Desenvolvedor do Amazon Simple Storage Service.

## Outros tipos de política

A AWS permite outros tipos de política capazes de definir o máximo de permissões concedidas por tipos de política mais comuns:

- Limites de permissões: definem o número máximo de permissões que uma política baseada em identidade pode conceder a uma entidade do IAM. Para saber mais sobre limites de permissões, consulte [Limites de permissões para identidades do IAM](#) no Guia do usuário do IAM.
- Políticas de Controle de Serviços (SCPs): as SCPs especificam o número máximo de permissões para uma organização ou uma unidade organizacional no AWS Organizations. Para saber mais, consulte [Políticas de controle de serviço](#) no Guia do usuário do AWS Organizations.
- Políticas de controle de recursos (RCPs): definem o número máximo de permissões disponíveis para recursos em suas contas. Consulte mais informações em [Resource control policies \(RCPs\)](#) no Guia do usuário do AWS Organizations.
- Políticas de sessão: políticas avançadas transmitidas como um parâmetro durante a criação de uma sessão temporária para um perfil ou um usuário federado. Para saber mais, consulte [Políticas de sessão](#) no Guia do usuário do IAM.

## Vários tipos de política

Quando vários tipos de política são aplicáveis a uma solicitação, é mais complicado compreender as permissões resultantes. Para saber como a AWS determina se deve permitir uma solicitação quando há vários tipos de política envolvidos, consulte [Lógica da avaliação de políticas](#) no Guia do Usuário do IAM.

# Como o Amazon API Gateway funciona com o IAM

Antes de usar o IAM para gerenciar o acesso ao API Gateway, você deve entender quais recursos do IAM estão disponíveis para uso com o API Gateway. Para obter uma visão de alto nível de como o API Gateway e outros serviços da AWS funcionam com o IAM, consulte [Serviços da AWS que funcionam com o IAM](#) no Guia do usuário do IAM.

## Tópicos

- [Políticas baseadas em identidade do API Gateway](#)
- [Políticas baseadas em recursos do API Gateway](#)
- [Autorização baseada em tags API Gateway](#)
- [Funções do IAM do API Gateway](#)

## Políticas baseadas em identidade do API Gateway

Com as políticas baseadas em identidade do IAM, é possível especificar quais ações e recursos são permitidos ou negados, bem como as condições sob as quais isso ocorre. O API Gateway oferece suporte a ações, recursos e chaves de condição específicos. Para obter mais informações sobre ações, recursos e chaves de condição específicos do API Gateway, consulte [Ações, recursos e chaves de condição do Amazon API Gateway Management](#) e [Ações, recursos e chaves de condição do Amazon API Gateway Management V2](#). Para obter informações sobre todos os elementos usados em uma política JSON, consulte [Referência de elementos de política JSON do IAM](#) no Guia do usuário do IAM.

O exemplo a seguir mostra uma política baseada em identidade que permite que um usuário crie ou atualize apenas APIs REST privadas. Para obter mais exemplos, consulte [the section called "Exemplos de políticas baseadas em identidade"](#).

## 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/*"
]
}
]
}

```

## Ações

O elemento `Action` de uma política JSON descreve as ações que você pode usar para permitir ou negar acesso em uma política.

As ações de política no API Gateway usam o seguinte prefixo antes da ação: `apigateway:`. As instruções de política devem incluir um elemento `Action` ou `NotAction`. O API Gateway define seu próprio conjunto de ações que descrevem as tarefas que podem ser executadas com esse serviço.

A expressão `Action` de gerenciamento de API está no formato `apigateway:action`, em que *action* representa uma das seguintes ações do API Gateway: GET, POST, PUT, DELETE, PATCH (para atualizar recursos) ou \*, que representa todas as ações anteriores.

Alguns exemplos da expressão `Action` incluem:

- **apigateway:\*** para todas as ações do API Gateway.
- **apigateway:GET** para apenas a ação GET no API Gateway.

Para especificar várias ações em uma única declaração, separe-as com vírgulas, conforme o seguinte:

```
"Action": [
 "apigateway:action1",
 "apigateway:action2"
```

Para obter informações sobre verbos a serem usados de HTTP para operações específicas do API Gateway, consulte a [Versão 1 da referência da API do Amazon API Gateway](#) (APIs REST) e a [Versão 2 da referência da API do Amazon API Gateway](#) (APIs WebSocket e HTTP).

Para obter mais informações, consulte [the section called “Exemplos de políticas baseadas em identidade”](#).

## Recursos

Os administradores podem usar as políticas JSON da AWS para especificar quem tem acesso a quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento de política JSON `Resource` especifica o objeto ou os objetos aos quais a ação se aplica. Como prática recomendada, especifique um recurso usando seu [nome do recurso da Amazon \(ARN\)](#). Para ações em que não é possível usar permissões em nível de recurso, use um curinga (\*) para indicar que a instrução se aplica a todos os recursos.

```
"Resource": "*"
```

Os recursos do API Gateway têm o seguinte formato ARN:

```
arn:aws:apigateway:region::resource-path-specifier
```

Por exemplo, para especificar uma API REST com o id *api-id* e seus sub-recursos, como autorizadores em sua declaração, use o ARN a seguir:

```
"Resource": "arn:aws:apigateway:us-east-2::/restapis/api-id/*"
```

Para especificar todas as APIs REST e sub-recursos que pertencem a uma conta específica, use o caractere curinga (\*):

```
"Resource": "arn:aws:apigateway:us-east-2::/restapis/*"
```

Para obter uma lista de tipos de recursos do API Gateway e seus ARNs, consulte [Referência de nome de recurso da Amazon \(ARN\) do API Gateway](#).

## Chaves de condição

Os administradores podem usar as políticas JSON AWS para especificar quem tem acesso a quê. Ou seja, qual entidade principal pode executar ações em quais recursos e em que condições.

O elemento `Condition` especifica quando as instruções são executadas com base em critérios definidos. É possível criar expressões condicionais que usem [agentes de condição](#), como “igual a” ou “menor que”, para fazer a condição da política corresponder aos valores na solicitação. Para ver todas as chaves de condição globais da AWS, consulte [Chaves de contexto de condição globais da AWS](#) no Guia do usuário do IAM.

O API Gateway define seu próprio conjunto de chaves de condição e também oferece suporte ao uso de algumas chaves de condição globais. Para obter uma lista de chaves de condição do API Gateway, consulte [Chaves de condição para o Amazon API Gateway](#) no Guia do usuário do IAM. Para obter informações sobre quais ações e recursos você pode usar uma chave de condição, consulte [Ações definidas pelo Amazon API Gateway](#).

Para obter informações sobre marcação, inclusive sobre o controle de acesso baseado em atributos, consulte [Atribuição de tags \(tagging\)](#).

## Exemplos

Para obter exemplos de políticas baseadas em identidade do API Gateway, consulte [Exemplos de políticas baseadas em identidade do Amazon API Gateway](#).

## Políticas baseadas em recursos do API Gateway

As políticas baseadas em recursos são documentos de políticas JSON que especificam quais ações um principal pode executar no recurso do API Gateway e sob quais condições. O API Gateway oferece suporte a políticas de permissões baseadas em recursos para APIs REST. Você usa políticas de recursos para controlar quem pode invocar uma API REST. Para obter mais informações, consulte [the section called “Usar políticas de recursos do API Gateway”](#).

## Exemplos

Para obter exemplos de políticas baseadas em recursos do API Gateway, consulte [Exemplos de política de recursos do API Gateway](#).

## Autorização baseada em tags API Gateway

Você pode anexar tags aos recursos do API Gateway ou transmitir tags em uma solicitação para o API Gateway. Para controlar o acesso baseado em tags, forneça informações sobre as tags no [elemento de condição](#) de uma política usando as chaves de condição `apigateway:ResourceTag/key-name`, `aws:RequestTag/key-name` ou `aws:TagKeys`. Para obter mais informações sobre como marcar recursos do API Gateway, consulte [the section called “Controle de acesso baseado em atributos”](#).

Para obter exemplos de políticas baseadas em identidade visando limitar o acesso a um recurso baseado nas tags desse recurso, consulte [Usar tags para controlar o acesso aos recursos da API REST do API Gateway](#).

## Funções do IAM do API Gateway

Uma [função do IAM](#) é uma entidade dentro da sua conta da AWS que tem permissões específicas.

### Usar credenciais temporárias com o API Gateway

É possível usar credenciais temporárias para fazer login com federação, assumir uma função do IAM ou assumir uma função entre contas. As credenciais de segurança temporárias são obtidas chamando operações da API do AWS STS, como [AssumeRole](#) ou [GetFederationToken](#).

O API Gateway oferece suporte ao uso de credenciais temporárias.

### Funções vinculadas ao serviço

[Perfis vinculados ao serviço](#) permitem que os serviços da AWS acessem recursos em outros serviços para concluir uma ação em seu nome. Os perfis vinculados a serviço aparecem em sua conta do IAM e são de propriedade do serviço. Um administrador do IAM pode visualizar, mas não pode editar as permissões para funções vinculadas ao serviço.

O API Gateway oferece suporte a funções vinculadas ao serviço. Para obter informações sobre como criar ou gerenciar funções vinculadas ao serviço do API Gateway, consulte [Usar funções vinculadas ao serviço para o API Gateway](#).

## Funções de serviço

Um serviço pode assumir uma [função de serviço](#) em seu nome. Uma função de serviço permite que o serviço acesse recursos em outros serviços para concluir uma ação em seu nome. Os perfis de serviço aparecem em sua conta do IAM e pertencem à conta, de modo que um administrador possa alterar as permissões para esse perfil. Porém, fazer isso pode alterar a funcionalidade do serviço.

O API Gateway oferece suporte a funções de serviço.

## Exemplos de políticas baseadas em identidade do Amazon API Gateway

Por padrão, os usuários e as funções do IAM não têm permissão para criar ou modificar recursos do API Gateway. Eles também não podem executar tarefas usando o Console de gerenciamento da AWS, a AWS CLI ou SDKs da AWS. Um administrador do IAM deve criar políticas do IAM que concedam aos usuários e funções permissão para executarem operações de API específicas nos recursos especificados de que precisam. O administrador deve anexar essas políticas aos usuários ou grupos do IAM que exigem essas permissões.

Para obter informações sobre como criar políticas do IAM, consulte [Como criar políticas na guia JSON](#) no Guia do usuário do IAM. Para obter informações sobre ações, recursos e condições específicas do API Gateway, consulte [Ações, recursos e chaves de condição do Amazon API Gateway Management](#) e [Ações, recursos e chaves de condição do Amazon API Gateway Management V2](#).

### Tópicos

- [Práticas recomendadas de política](#)
- [Permitir que os usuários visualizem suas próprias permissões](#)
- [Permissões de leitura simples](#)
- [Crie apenas os autorizadores REQUEST ou JWT](#)
- [Exija que o endpoint padrão execute-api esteja desabilitado](#)
- [Permita que usuários criem ou atualizem apenas APIs REST privadas](#)
- [Exija que as rotas da API tenham autorização](#)
- [Impeça que um usuário crie ou atualize um link da VPC](#)
- [Exemplos de políticas para usar regras de roteamento](#)

## Práticas recomendadas de política

As políticas baseadas em identidade determinam se alguém pode criar, acessar ou excluir recursos do API Gateway em sua conta. Essas ações podem incorrer em custos para sua Conta da AWS. Ao criar ou editar políticas baseadas em identidade, siga estas diretrizes e recomendações:

- Comece com as políticas gerenciadas pela AWS e avance para as permissões de privilégio mínimo: para começar a conceder permissões a seus usuários e workloads, use as políticas gerenciadas pela AWS, que concedem permissões para muitos casos de uso comuns. Elas estão disponíveis em sua Conta da AWS. Recomendamos que você reduza ainda mais as permissões definindo políticas gerenciadas pelo cliente da AWS que são específicas para seus casos de uso. Para saber mais, consulte [Políticas gerenciadas pela AWS](#) ou [Políticas gerenciadas pela AWS para funções de trabalho](#) no Guia do usuário do IAM.
- Aplique permissões de privilégio mínimo: ao definir permissões com as políticas do IAM, conceda apenas as permissões necessárias para executar uma tarefa. Você faz isso definindo as ações que podem ser executadas em recursos específicos sob condições específicas, também conhecidas como permissões de privilégio mínimo. Para saber mais sobre como usar o IAM para aplicar permissões, consulte [Políticas e permissões no IAM](#) no Guia do usuário do IAM.
- Use condições nas políticas do IAM para restringir ainda mais o acesso: é possível adicionar uma condição às políticas para limitar o acesso a ações e recursos. Por exemplo, é possível escrever uma condição de política para especificar que todas as solicitações devem ser enviadas usando SSL. Também pode usar condições para conceder acesso a ações de serviço, se elas forem usadas por meio de um AWS service (Serviço da AWS) específico, como o CloudFormation. Para saber mais, consulte [Elementos da política JSON do IAM: condição](#) no Guia do usuário do IAM.
- Use o IAM Access Analyzer para validar suas políticas do IAM a fim de garantir permissões seguras e funcionais: o IAM Access Analyzer valida as políticas novas e existentes para que elas sigam a linguagem de política do IAM (JSON) e as práticas recomendadas do IAM. O IAM Access Analyzer oferece mais de cem verificações de política e recomendações práticas para ajudar a criar políticas seguras e funcionais. Para saber mais, consulte [Validação de políticas do IAM Access Analyzer](#) no Guia do Usuário do IAM.
- Exigir autenticação multifator (MFA): se houver um cenário que exija usuários do IAM ou um usuário raiz em sua Conta da AWS, ative a MFA para obter segurança adicional. Para exigir MFA quando as operações de API forem chamadas, adicione condições de MFA às suas políticas. Para saber mais, consulte [Configuração de acesso à API protegido por MFA](#) no Guia do Usuário do IAM.



Para saber mais sobre as práticas recomendadas do IAM, consulte [Práticas recomendadas de segurança no IAM](#) no Guia do usuário do IAM.

## Permitir que os usuários visualizem suas próprias permissões

Este exemplo mostra como criar uma política que permita que os usuários do IAM visualizem as políticas gerenciadas e em linha anexadas a sua identidade de usuário. Essa política inclui permissões para concluir essa ação no console ou de forma programática usando a AWS CLI ou a API da AWS.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "ViewOwnUserInfo",
 "Effect": "Allow",
 "Action": [
 "iam:GetUserPolicy",
 "iam:ListGroupsWithUser",
 "iam:ListAttachedUserPolicies",
 "iam:ListUserPolicies",
 "iam:GetUser"
],
 "Resource": ["arn:aws:iam::*:user/${aws:username}"]
 },
 {
 "Sid": "NavigateInConsole",
 "Effect": "Allow",
 "Action": [
 "iam:GetGroupPolicy",
 "iam:GetPolicyVersion",
 "iam:GetPolicy",
 "iam:ListAttachedGroupPolicies",
 "iam:ListGroupPolicies",
 "iam:ListPolicyVersions",
 "iam:ListPolicies",
 "iam:ListUsers"
],
 "Resource": "*"
 }
]
}
```

## Permissões de leitura simples

Esta política de exemplo concede permissão ao usuário para obter informações sobre todos os recursos de uma API HTTP ou WebSocket com o identificador de a123456789 na região us-east-1 da AWS. O recurso `arn:aws:apigateway:us-east-1::/apis/a123456789/*` inclui todos os sub-recursos da API, como autorizadores e implantações.

### JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "apigateway:GET"
],
 "Resource": [
 "arn:aws:apigateway:us-east-1::/apis/a123456789/*"
]
 }
]
}
```

## Crie apenas os autorizadores REQUEST ou JWT

Esta política de exemplo permite que um usuário crie APIs apenas com os autorizadores REQUEST ou JWT, inclusive por meio da [importação](#). Na seção Resource da política, o `arn:aws:apigateway:us-east-1::/apis/??????????` exige que os recursos tenham um máximo de dez caracteres, o que exclui os sub-recursos de uma API. Este exemplo usa `ForAllValues` na seção Condition porque os usuários podem criar vários autorizadores de uma só vez importando uma 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"
]
 }
 }
}

```

Exija que o endpoint padrão **execute-api** esteja desabilitado

Esta política de exemplo permite que os usuários criem, atualizem ou importem uma API, com a exigência de que o `DisableExecuteApiEndpoint` seja `true`. Quando o `DisableExecuteApiEndpoint` for `true`, os clientes não poderão usar o endpoint padrão `execute-api` para invocar uma API.

Nós usamos a condição `BoolIfExists` para lidar com uma chamada para atualizar uma API que não tenha a chave de condição `DisableExecuteApiEndpoint` preenchida. Quando um usuário tenta criar ou importar uma API, a chave de condição `DisableExecuteApiEndpoint` é sempre preenchida.

Como o recurso `apis/*` também captura sub-recursos, como autorizadores ou métodos, nós o estendemos explicitamente apenas para APIs com uma declaração `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/*/*"
]
 }
]
}
```

## Permita que usuários criem ou atualizem apenas APIs REST privadas

Esta política de exemplo usa chaves de condição para exigir que um usuário crie somente APIs PRIVATE e para evitar atualizações que possam alterar uma API PRIVATE para outro tipo, como REGIONAL.

Usamos `ForAllValues` para exigir que cada `EndpointType` adicionado a uma API seja PRIVATE. Usamos uma chave de condição de recurso para permitir qualquer atualização em uma API, desde que seja PRIVATE. `ForAllValues` aplica-se somente quando uma chave de condição está presente.

Nós usamos a correspondência não greedy (?) para corresponder explicitamente aos IDs de API para evitar a permissão de recursos que não sejam APIs, como, por exemplo, 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/*"
]
}
]
}

```

## Exija que as rotas da API tenham autorização

Esta política faz com que as tentativas de criação ou atualização de uma rota (inclusive por meio da [importação](#)) falhem se a rota não tiver autorização. O `ForAnyValue` considera a tentativa falsa se a chave não estiver presente, como, por exemplo, quando uma rota não está sendo criada ou atualizada. Nós usamos `ForAnyValue` porque várias rotas podem ser criadas por meio da importação.

### 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"
 }
 }
 }
]
}

```

## Impeça que um usuário crie ou atualize um link da VPC

Esta política impede que um usuário crie ou atualize um link da VPC. Um link da VPC permite que você exponha recursos em uma Amazon VPC para clientes fora da 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/*"
]
 }
]
}
```

## Exemplos de políticas para usar regras de roteamento

As políticas de exemplo a seguir mostram como usar as chaves de condição `RoutingRule` para controlar como os usuários podem rotear o tráfego de seus nomes de domínio personalizados para suas APIs REST. Você pode usar esses exemplos para criar políticas granulares para o tipo de regras de roteamento que os usuários podem criar. Para obter mais informações, consulte [the section called “Regras de roteamento”](#).

Impedir que um usuário altere a forma como um nome de domínio personalizado encaminha uma solicitação

Essa política impede que um usuário crie ou atualize `BasePathMapping`, `ApiMapping` ou `RoutingRule`. Todos esses recursos podem mudar a maneira como um nome de domínio personalizado encaminha solicitações para as APIs.

## JSON

```
{
```



```

"Version": "2012-10-17",
"Statement": [
 {
 "Sid": "DenyAccessBasePathMappingsApiMappingsRoutingRules",
 "Effect": "Deny",
 "Action": "apigateway:*",
 "Resource": [
 "arn:aws:apigateway:us-east-1::/domainnames/example.com/basepathmappings/*",
 "arn:aws:apigateway:us-east-1::/domainnames/example.com/apimappings/*",
 "arn:aws:apigateway:us-east-1:111122223333:/domainnames/example.com/routingrules/*"
]
 }
]
}

```

Permitir que um usuário atualize uma regra de roteamento para determinadas prioridades

Essa política permite que um usuário atualize somente uma regra de roteamento para uma prioridade entre 1001 e 2000. Você pode usar essa regra para separar suas regras de produção das regras de menor prioridade e, em seguida, permitir que os usuários modifiquem as regras de menor prioridade sem afetar as regras de produção.

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
 }
}
]
}

```

Permitir que um usuário atualize uma regra de roteamento ou mapeamento do caminho base para um determinado valor do caminho base

Essa política permite que um usuário atualize somente um mapeamento de caminho base para qualquer caminho base que comece com `orders` ou atualize uma regra de roteamento correspondente a um caminho base que comece com `orders`. Nessa política, um usuário pode atualizar um mapeamento de caminho base ou regra de roteamento para `orders/create` ou `orders123`, mas não `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"
 }
 }
 }
]
}

```

```
}
```

Permitir que um usuário atualize o modo de roteamento para valores específicos

Essa política permite que um usuário atualize somente o modo de roteamento para `API_MAPPING_ONLY` e `ROUTING_RULE_THEN_API_MAPPING`. Para obter mais informações sobre o modo de roteamento, consulte [the section called “Definir o modo de roteamento para o nome de domínio 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*"
 }
 }
 }
]
}
```

## Exemplos de políticas baseadas em recursos do Amazon API Gateway

Para obter exemplos de política baseada em recursos, consulte [the section called “Exemplos de política de recursos do API Gateway”](#).

## Solução de problemas de identidade e acesso do Amazon API Gateway

Use as informações a seguir para ajudar a diagnosticar e corrigir problemas comuns que você possa encontrar ao trabalhar com o API Gateway e o IAM.

Tópicos

- [Não tenho autorização para executar uma ação no API Gateway](#)
- [Não estou autorizado a executar iam:PassRole](#)
- [Desejo permitir que pessoas fora da minha conta da AWS acessem meus recursos do API Gateway](#)

## Não tenho autorização para executar uma ação no API Gateway

Se você receber uma mensagem de erro informando que não tem autorização para executar uma ação, suas políticas deverão ser atualizadas para permitir que você realize a ação.

O erro do exemplo a seguir ocorre quando o usuário do IAM `mateojackson` tenta usar o console para visualizar detalhes sobre um atributo `my-example-widget` fictício, mas não tem as permissões `apigateway:GetWidget` fictícias.

```
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
```

Nesse caso, a política do usuário `mateojackson` deve ser atualizada para permitir o acesso ao recurso `my-example-widget` usando a ação `apigateway:GetWidget`.

Se você precisar de ajuda, entre em contato com seu administrador AWS. Seu administrador é a pessoa que forneceu suas credenciais de login.

## Não estou autorizado a executar iam:PassRole

Se receber uma mensagem de erro informando que você não está autorizado a realizar a ação `iam:PassRole`, suas políticas deverão ser atualizadas para permitir a transmissão de um perfil para o API Gateway.

Alguns Serviços da AWS permitem que você transmita um perfil existente para o serviço, em vez de criar um perfil de serviço ou um perfil vinculado ao serviço. Para fazer isso, é preciso ter permissões para passar o perfil para o serviço.

O erro exemplificado a seguir ocorre quando uma usuária do IAM chamada `marymajor` tenta usar o console para executar uma ação no API Gateway. No entanto, a ação exige que o serviço tenha permissões concedidas por um perfil de serviço. Mary não tem permissões para passar o perfil para o serviço.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

Nesse caso, as políticas de Mary devem ser atualizadas para permitir que ela realize a ação `iam:PassRole`.

Se você precisar de ajuda, entre em contato com seu administrador AWS. Seu administrador é a pessoa que forneceu suas credenciais de login.

## Desejo permitir que pessoas fora da minha conta da AWS acessem meus recursos do API Gateway

Você pode criar uma função que os usuários de outras contas ou pessoas fora da sua organização podem usar para acessar seus recursos. É possível especificar quem é confiável para assumir o perfil. Para serviços que oferecem compatibilidade com políticas baseadas em recursos ou listas de controle de acesso (ACLs), é possível usar essas políticas para conceder às pessoas acesso aos seus recursos.

Para saber mais, consulte o seguinte:

- Para saber se o API Gateway oferece suporte a esses recursos, consulte [Como o Amazon API Gateway funciona com o IAM](#).
- Para saber como conceder acesso a seus recursos em todas as Contas da AWS pertencentes a você, consulte [Fornecimento de acesso a um usuário do IAM em outra Conta da AWS pertencente a você](#) no Guia de usuário do IAM.
- Para saber como conceder acesso a seus recursos para Contas da AWS de terceiros, consulte [Fornecimento de acesso a Contas da AWS pertencentes a terceiros](#) no Guia do usuário do IAM.
- Para saber como conceder acesso por meio da federação de identidades, consulte [Conceder acesso a usuários autenticados externamente \(federação de identidades\)](#) no Guia do usuário do IAM.
- Para conhecer a diferença entre perfis e políticas baseadas em recurso para acesso entre contas, consulte [Acesso a recursos entre contas no IAM](#) no Guia do usuário do IAM.

## Usar funções vinculadas ao serviço para o API Gateway

O Amazon API Gateway usa [funções vinculadas ao serviço](#) do AWS Identity and Access Management (IAM). A função vinculada ao serviço é um tipo exclusivo de função do IAM vinculada

diretamente ao API Gateway. As funções vinculadas a serviços são predefinidas pelo API Gateway e incluem todas as permissões que o serviço requer para chamar outros serviços da AWS em seu nome.

Uma função vinculada ao serviço facilita a configuração do API Gateway, já que não é preciso adicionar as permissões necessárias manualmente. O API Gateway define as permissões das funções vinculadas a serviços e, exceto se definido de outra forma, somente o API Gateway pode assumir suas funções. As permissões definidas incluem a política de confiança e a política de permissões, e essa política não pode ser anexada a nenhuma outra entidade do IAM.

Você pode excluir uma função vinculada ao serviço somente depois de excluir os recursos relacionados da. Isso protege os recursos do API Gateway, pois não é possível remover por engano as permissões para acessar os recursos.

Para obter informações sobre outros serviços compatíveis com perfis vinculados a serviços, consulte [Serviços da AWS compatíveis com o IAM](#) e procure os serviços que contenham Sim na coluna Perfil vinculado a serviço. Escolha um Sim com um link para exibir a documentação da função vinculada a serviço desse serviço.

## Permissões de função vinculada ao serviço para o API Gateway

O API Gateway usa o perfil vinculado ao serviço chamado `AWSServiceRoleForAPIGateway`: permite que o API Gateway acesse o Elastic Load Balancing, o Amazon Data Firehose e outros recursos de serviços em seu nome.

A função vinculada ao serviço `AWSServiceRoleForAPIGateway` confia nos seguintes serviços para assumir a função:

- `ops.apigateway.amazonaws.com`

A política de permissões da função permite que o API Gateway conclua as seguintes ações nos 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/*"
 }
]

```



```
}
```

Você deve configurar permissões para que uma entidade do IAM (por exemplo, um usuário, grupo ou função) crie, edite ou exclua uma função vinculada ao serviço. Para obter mais informações, consulte [Permissões de função vinculada ao serviço](#) no Guia do usuário do IAM.

## Criar uma função vinculada ao serviço para o API Gateway

Você não precisa criar manualmente uma função vinculada a serviço. Quando você cria uma API, um nome de domínio personalizado ou um link de VPC no Console de gerenciamento da AWS, a AWS CLI ou a API da AWS, o API Gateway cria a função vinculada ao serviço para você.

Se você excluir essa função vinculada ao serviço e precisar criá-la novamente, poderá usar esse mesmo processo para recriar a função em sua conta. Quando você cria uma API, um nome de domínio personalizado ou um link de VPC, o API Gateway cria a função vinculada ao serviço para você novamente.

## Editar uma função vinculada ao serviço para o API Gateway

O API Gateway não permite que você edite a função vinculada ao serviço `AWSServiceRoleForAPIGateway`. Depois que você criar uma função vinculada a serviço, não poderá alterar o nome da função, pois várias entidades podem fazer referência à função. No entanto, você poderá editar a descrição do perfil usando o IAM. Para obter mais informações, consulte [Editar uma função vinculada ao serviço](#) no Guia do usuário do IAM.

## Excluir uma função vinculada ao serviço para o API Gateway

Se você não precisar mais usar um recurso ou serviço que requer uma função vinculada a serviço, é recomendável excluí-la. Dessa forma, você não tem uma entidade não utilizada que não seja monitorada ativamente ou mantida. No entanto, você deve limpar os recursos de sua função vinculada ao serviço antes de excluí-la manualmente.

### Note

Se o serviço API Gateway estiver usando a função quando você tentar excluir os recursos, a exclusão poderá falhar. Se isso acontecer, espere alguns minutos e tente a operação novamente.

## Como excluir recursos do API Gateway usados pela AWSServiceRoleForAPIGateway

1. Abra o console do API Gateway em <https://console.aws.amazon.com/apigateway/>.
2. Navegue para a API, o nome de domínio personalizado ou o link da VPC que usa a função vinculada ao serviço.
3. Use o console para excluir o recurso.
4. Repita o procedimento para excluir todas as APIs, nomes de domínio personalizados ou links de VPC que usam a função vinculada ao serviço.

## Como excluir manualmente o perfil vinculado ao serviço usando o IAM

Use o console do IAM, a AWS CLI ou a API da AWS para excluir a função vinculada ao serviço AWSServiceRoleForAPIGateway. Para obter mais informações, consulte [Excluir uma função vinculada ao serviço](#) no Guia do usuário do IAM.

## Regiões compatíveis com funções vinculadas a serviços do API Gateway

O API Gateway oferece suporte a funções vinculadas a serviços em todas as regiões em que o serviço está disponível. Para obter mais informações, consulte [Endpoints de serviço do AWS](#).

## Atualizações do API Gateway para políticas gerenciadas da AWS

Visualize detalhes sobre atualizações em políticas gerenciadas pela AWS para o API Gateway desde que este serviço começou a rastrear essas alterações. Para obter alertas automáticos sobre alterações feitas nesta página, inscreva-se no feed RSS na página [Histórico de documentos](#) do API Gateway.

| Alteração                                                                                                 | Descrição                                                                                                                                | Data                |
|-----------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|---------------------|
| Adição de suporte a <code>acm:GetCertificate</code> à política <code>AWSServiceRoleForAPIGateway</code> . | A política <code>AWSServiceRoleForAPIGateway</code> agora inclui permissão para chamar a ação de API <code>GetCertificate</code> do ACM. | 12 de julho de 2021 |

| Alteração                                     | Descrição                                                                          | Data                |
|-----------------------------------------------|------------------------------------------------------------------------------------|---------------------|
| API Gateway começou a controlar as alterações | API Gateway começou a controlar alterações para suas políticas gerenciadas da AWS. | 12 de julho de 2021 |

## Registro em log e monitoramento no Amazon API Gateway

O monitoramento é uma parte importante para manter a confiabilidade, a disponibilidade e a performance do API Gateway e das suas soluções da AWS. É necessário coletar dados de monitoramento de todas as partes de sua solução da AWS para depurar uma falha de vários pontos com mais facilidade, caso ocorra. A AWS fornece várias ferramentas para monitorar seus recursos do e responder a possíveis incidentes:

### Amazon CloudWatch Logs

Para ajudar a depurar problemas relacionados à execução de solicitação ou ao acesso do cliente à sua API, é possível permitir que o CloudWatch Logs registre chamadas de API. Para obter mais informações, consulte [the section called “Logs do CloudWatch”](#).

### Alarmes do Amazon CloudWatch

Ao usar alarmes do CloudWatch, você observa uma única métrica durante um período especificado. Se a métrica exceder determinado limite, uma notificação será enviada para um tópico do Amazon Simple Notification Service ou para uma política do AWS Auto Scaling. Os alarmes do CloudWatch não invocam ações quando uma métrica está em um estado específico. O estado deve ter sido alterado e mantido por uma quantidade especificada de períodos. Para ter mais informações, consulte [the section called “Métricas do CloudWatch”](#).

### Registro em log de acesso ao Firehose

Para ajudar a depurar problemas relacionados ao acesso do cliente à sua API, é possível permitir que o Firehose registre em log chamadas de API. Para ter mais informações, consulte [the section called “Firehose”](#).

### AWS CloudTrail

O CloudTrail fornece um registro de ações executadas por um usuário, uma função ou um serviço da AWS no API Gateway. Usando as informações coletadas pelo CloudTrail, é possível determinar a solicitação feita para o API Gateway, o endereço IP do qual a solicitação foi feita,

quem fez a solicitação, quando ela foi feita e detalhes adicionais. Para obter mais informações, consulte [the section called “Trabalhar com o CloudTrail”](#).

## AWS X-Ray

O X-Ray é um produto da AWS que coleta dados sobre as solicitações atendidas pela aplicação e os usa para construir um mapa de serviços que pode ser usado para identificar problemas na aplicação e oportunidades de otimização. Para obter mais informações, consulte [the section called “Configurar o AWS X-Ray”](#).

## AWS Config

O AWS Config fornece uma visão detalhada da configuração dos recursos da AWS na conta. Você pode ver como os recursos estão relacionados, obter um histórico de alterações de configuração e ver como os relacionamentos e as configurações foram alterados ao longo do tempo. É possível usar o AWS Config para definir regras que avaliam configurações de recursos para conformidade de dados. As regras do AWS Config representam as definições de configuração ideais para os recursos do API Gateway. Se um recurso violar uma regra e for sinalizado como não compatível, o AWS Config poderá alertá-lo usando um tópico do Amazon Simple Notification Service (Amazon SNS). Para obter mais detalhes, consulte [the section called “Trabalhar com o AWS Config”](#).

## Registrar em log chamadas de APIs do Amazon API Gateway usando o AWS CloudTrail

O Amazon API Gateway é integrado ao [AWS CloudTrail](#), um serviço que fornece um registro das ações realizadas por um usuário, por um perfil ou por um AWS service (Serviço da AWS). O CloudTrail captura as chamadas de API REST do serviço do API Gateway como eventos. As chamadas capturadas incluem as chamadas do console do API Gateway e as chamadas de código para as APIs do serviço do API Gateway. Ao fazer uso das informações coletadas pelo CloudTrail, é possível determinar a solicitação feita ao API Gateway, o endereço IP no qual a solicitação foi feita, quando a solicitação foi feita e detalhes adicionais.

### Note

[TestInvokeAuthorizer](#) e [TestInvokeMethod](#) não são registrados no CloudTrail.

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar o seguinte:

- Se a solicitação foi feita com credenciais de usuário raiz ou credenciais de usuário.
- Se a solicitação foi feita em nome de um usuário do Centro de Identidade do IAM.
- Se a solicitação foi feita com credenciais de segurança temporárias de um perfil ou de um usuário federado.
- Se a solicitação foi feita por outro AWS service (Serviço da AWS).

Quando a Conta da AWS é criada, o CloudTrail torna-se ativo nela e você tem acesso automático ao Histórico de eventos do CloudTrail. O Histórico de eventos do CloudTrail fornece um registro visualizável, pesquisável, baixável e imutável dos últimos 90 dias de eventos de gerenciamento gravados em uma Região da AWS. Para obter mais informações, consulte [Trabalhar com histórico de eventos do CloudTrail](#) no Guia do usuário do AWS CloudTrail. Não há cobranças do CloudTrail pela visualização do Histórico de eventos.

Para obter um registro contínuo de eventos em sua Conta da AWS nos últimos 90 dias, crie uma trilha ou um armazenamento de dados de eventos do [CloudTrail Lake](#).

## Trilhas do CloudTrail

Uma trilha permite que o CloudTrail entregue arquivos de log a um bucket do Amazon S3. As trilhas criadas usando o Console de gerenciamento da AWS são de várias regiões. Só é possível criar uma trilha de região única ou de várias regiões usando a AWS CLI. Criar uma trilha de várias regiões é uma prática recomendada, pois você captura atividades em todas as Regiões da AWS da conta. Ao criar uma trilha de região única, é possível visualizar somente os eventos registrados na Região da AWS da trilha. Para obter mais informações sobre trilhas, consulte [Criar uma trilha para a Conta da AWS](#) e [Criar uma trilha para uma organização](#) no Guia do usuário do AWS CloudTrail.

Uma cópia dos eventos de gerenciamento em andamento pode ser entregue no bucket do Amazon S3 sem nenhum custo via CloudTrail com a criação de uma trilha; no entanto, há cobranças de armazenamento do Amazon S3. Para obter mais informações sobre os preços do CloudTrail, consulte [Preços do AWS CloudTrail](#). Para receber informações sobre a definição de preços do Amazon S3, consulte [Definição de preços do Amazon S3](#).

## Armazenamentos de dados de eventos do CloudTrail Lake

O CloudTrail Lake permite executar consultas baseadas em SQL nos eventos. O CloudTrail Lake converte eventos existentes em formato JSON baseado em linhas para o formato [Apache ORC](#). O ORC é um formato colunar de armazenamento otimizado para recuperação rápida de dados. Os eventos são agregados em armazenamentos de dados de eventos, que são coleções imutáveis de eventos baseados nos critérios selecionados com a aplicação de [seletores de eventos avançados](#). Os seletores que aplicados a um armazenamento de dados de eventos controlam quais eventos persistem e estão disponíveis para consulta. Para obter mais informações sobre o CloudTrail Lake, consulte [Trabalhar com o AWS CloudTrail Lake](#), no Guia do usuário do AWS CloudTrail.

Os armazenamentos de dados de eventos e consultas do CloudTrail Lake incorrem em custos. Ao criar um armazenamento de dados de eventos, você escolhe a [opção de preço](#) que deseja usar para ele. A opção de preço determina o custo para a ingestão e para o armazenamento de eventos, e o período de retenção padrão e máximo para o armazenamento de dados de eventos. Para saber mais sobre os preços do CloudTrail, consulte [Preços do AWS CloudTrail](#).

## Eventos de gerenciamento do API Gateway no CloudTrail

Os [eventos de gerenciamento](#) fornecem informações sobre operações de gerenciamento executadas em recursos na Conta da AWS. Elas também são conhecidas como operações de ambiente de gerenciamento. Por padrão, o CloudTrail registra eventos de gerenciamento em logs.

O Amazon API Gateway registra em log todas as ações do API Gateway como eventos de gerenciamento, exceto [TestInvokeAuthorizer](#) e [TestInvokeMethod](#). Para encontrar uma lista das ações do Amazon API Gateway que o API Gateway registra em log no CloudTrail, consulte [Amazon API Gateway API Reference](#).

## Exemplo de evento do API Gateway

Um evento representa uma única solicitação de qualquer origem e inclui informações sobre a operação solicitada, a data e a hora da operação da API, os parâmetros de solicitação etc. Os arquivos de log do CloudTrail não são um rastreamento de pilha ordenada de chamadas de API públicas, portanto não são exibidos em uma ordem específica.

O seguinte exemplo mostra uma evento do CloudTrail que demonstra a ação `GetResource` do 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 obter informações sobre o conteúdo dos registros do CloudTrail, consulte [Conteúdo dos registros do CloudTrail](#) no Guia do usuário do AWS CloudTrail.

## Monitorar a configuração da API do API Gateway com AWS Config

Você pode usar o [AWS Config](#) para registrar as alterações de configuração feitas em seus recursos de API do API Gateway e enviar notificações com base em alterações de recursos. A manutenção de um histórico de alteração de configurações do API Gateway é útil para casos de uso de solução de problemas operacionais, auditoria e conformidade.

AWS ConfigO pode rastrear alterações em:

- Configuração de estágio de API, como:
  - configurações de cluster de cache
  - configurações de limites
  - configurações de logs de acesso
  - a implantação ativa definida no estágio
- Configuração de API, como:
  - configuração do endpoint
  - versão
  - protocolo
  - tags

Além disso, o recurso do Regras do AWS Config permite que você defina regras de configuração e detecte, rastreie e alerte violações dessas regras automaticamente. Além disso, o rastreamento de alterações nessas propriedades de configuração de recursos permite criar regras do AWS Config acionadas por alterações para recursos do API Gateway e testar as configurações de recursos em relação às melhores práticas.

Você pode habilitar o AWS Config em sua conta usando o console do AWS Config ou a AWS CLI. Selecione os tipos de recurso para os quais você deseja rastrear as alterações. Se você já configurou o AWS Config para registrar todos os tipos de recursos, esses recursos do API Gateway serão registrados automaticamente na sua conta. O suporte ao Amazon API Gateway em AWS Config está disponível em todas as regiões públicas da AWS e em AWS GovCloud (US). Para obter a lista completa de regiões compatíveis, consulte [Endpoints e cotas do Amazon API Gateway](#) na Referência geral da AWS.

### Tópicos

- [Tipos de recursos compatíveis](#)



- [Configurar o AWS Config](#)
- [Configuração do AWS Config para registrar recursos do API Gateway](#)
- [Vizualização de detalhes de configuração do API Gateway no console do AWS Config](#)
- [Avaliação de recursos do API Gateway usando regras do AWS Config](#)

## Tipos de recursos compatíveis

Os seguintes tipos de recursos do API Gateway são integrados ao AWS Config e documentados em [Relacionamentos de recursos e tipos de recursos da AWS com suporte do AWS Config](#):

- `AWS::ApiGatewayV2::Api` (WebSocket e API HTTP)
- `AWS::ApiGateway::RestApi` (API REST)
- `AWS::ApiGatewayV2::Stage` (estágio de API WebSocket e HTTP)
- `AWS::ApiGateway::Stage` (Estágio de API REST)

Para obter mais informações sobre o AWS Config, consulte o [Guia do desenvolvedor do AWS Config](#). Para obter informações sobre a definição de preço, consulte a [página de informações sobre a definição de preço do AWS Config](#).

### Important

Se você alterar qualquer uma das seguintes propriedades da API depois que a API for implantada, deverá [reimplantá-la](#) para propagar as alterações. Caso contrário, você verá as alterações de atributo no console do AWS Config, mas as configurações de propriedade anteriores ainda estarão em vigor. O comportamento de tempo de execução da API permanecerá inalterado.

- **`AWS::ApiGateway::RestApi`** – `binaryMediaTypes`, `minimumCompressionSize`, `apiKeySource`
- **`AWS::ApiGatewayV2::Api`** – `apiKeySelectionExpression`

## Configurar o AWS Config

Para fazer a configuração inicial do AWS Config, consulte os tópicos a seguir no [Guia do desenvolvedor do AWS Config](#).

- [Configuração do AWS Config no console](#)
- [Configuração do AWS Config com a AWS CLI](#)

## Configuração do AWS Config para registrar recursos do API Gateway

Por padrão, o AWS Config registra as alterações de configuração para todos os tipos de recursos regionais com suporte, que ele descobre na região em que seu ambiente está sendo executado. Você pode personalizar o AWS Config para registrar alterações somente para tipos de recursos específicos ou alterações em recursos globais.

Para saber mais sobre os recursos regionais versus globais e saber como personalizar sua configuração do AWS Config, consulte [Selecionar quais recursos o AWS Config registra](#).

## Vizualização de detalhes de configuração do API Gateway no console do AWS Config

É possível usar o console do AWS Config para procurar recursos do API Gateway atuais e históricos e obter detalhes sobre suas configurações. O procedimento a seguir mostra como encontrar informações sobre uma API do API Gateway.

Como localizar um recurso do API Gateway no console de configuração do AWS

1. Abra o [console do AWS Config](#).
2. Escolha Resources (Recursos).
3. Na página de inventário Resource (Recurso), selecione Resources (Recursos).
4. Abra o menu Resource type (Tipo de recurso), role a tela até APIGateway ou APIGatewayV2 e selecione um ou mais tipos de recursos do API Gateway.
5. Escolha Look up.
6. Selecione um ID de recurso na lista de recursos que o AWS Config exibe. O AWS Config exibe detalhes de configuração e outras informações sobre o recurso que você selecionou.
7. Para ver os detalhes completos da configuração registrada, selecione View Details (Exibir detalhes).

Para saber mais maneiras de localizar um recurso e exibir informações nesta página, consulte [Visualizar configurações de recursos e histórico da AWS](#) no Guia do desenvolvedor do AWS Config.

## Avaliação de recursos do API Gateway usando regras do AWS Config

É possível criar regras do AWS Config que representam as definições de configuração ideais para seus recursos do API Gateway. Você pode usar [Regras de configuração gerenciadas pelo AWS Config](#) predefinidas ou definir regras personalizadas. O AWS Config monitora de forma contínua as alterações na configuração dos seus recursos para determinar se essas alterações violam alguma condição em suas regras. O console AWS Config mostra o status de compatibilidade de suas regras e recursos.

Se um recurso violar uma regra e for sinalizado como não compatível, o AWS Config poderá alertá-lo usando um tópico do [Guia do desenvolvedor do Amazon Simple Notification](#) (Amazon SNS). Para consumir programaticamente os dados nesses alertas do AWS Config, use uma fila do Amazon Simple Queue Service (Amazon SQS) como o endpoint de notificação para o tópico do Amazon SNS.

Para saber mais sobre como configurar e usar regras, consulte [Avaliação de recursos com regras](#) no [Guia do desenvolvedor do AWS Config](#).

## Validação de conformidade do Amazon API Gateway

Para saber se um AWS service (Serviço da AWS) está no escopo de programas de conformidade específicos, consulte [Serviços da AWS no escopo por programa de conformidade](#) e selecione o programa de conformidade em que você está interessado. Para obter informações gerais, consulte [Programas de Conformidade da AWS](#).

É possível baixar relatórios de auditoria de terceiros usando o AWS Artifact. Para saber mais, consulte [Baixar relatórios no AWS Artifact](#).

Sua responsabilidade de conformidade ao usar o Serviços da AWS é determinada pela confidencialidade dos seus dados, pelos objetivos de conformidade da sua empresa e pelos regulamentos e leis aplicáveis. Para ter mais informações sobre sua responsabilidade de conformidade ao usar Serviços da AWS, consulte [Documentação de segurança da AWS](#).

## Resiliência no Amazon API Gateway

A infraestrutura global da AWS é criada com base em regiões da AWS e zonas de disponibilidade. As regiões da AWS As regiões fornecem várias zonas de disponibilidade separadas e isoladas fisicamente, conectadas com baixa latência, throughput elevado e redes altamente redundantes.

Com as zonas de disponibilidade, é possível projetar e operar aplicativos e bancos de dados que automaticamente executam o failover entre as zonas sem interrupção. As zonas de disponibilidade são altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de data centers tradicionais.

Como um serviço regional totalmente gerenciado, o API Gateway opera em várias zonas de disponibilidade em cada região, usando a redundância das zonas de disponibilidade para minimizar falhas na infraestrutura como uma categoria de risco de disponibilidade. O API Gateway foi projetado para se recuperar automaticamente da falha de uma zona de disponibilidade.

Para mais informações sobre regiões e zonas de disponibilidade da AWS, consulte [Infraestrutura global da AWS](#).

Para evitar que suas APIs fiquem sobrecarregadas com muitas solicitações, o API Gateway limita as solicitações para suas APIs. Especificamente, o API Gateway define um limite para uma taxa de estado fixo e uma intermitência de envios de solicitações para todas as APIs na sua conta. É possível configurar a limitação personalizada para suas APIs. Para saber mais, consulte [Controle de utilização das solicitações das APIs REST para ter um melhor throughput no API Gateway](#).

É possível usar as verificações de integridade do Route 53 para controlar o failover de DNS de uma API do API Gateway em uma região primária para uma API do API Gateway em uma região secundária. Para ver um exemplo, consulte [the section called “failover de DNS”](#).

## Segurança de infraestrutura no Amazon API Gateway

Por ser um serviço gerenciado, o Amazon API Gateway é protegido pela segurança da rede global da AWS. Para obter informações sobre serviços de segurança da AWS e como a AWS protege a infraestrutura, consulte [Segurança na Nuvem AWS](#). Para projetar seu ambiente da AWS usando as práticas recomendadas de segurança da infraestrutura, consulte [Proteção de Infraestrutura](#) em Pilar de Segurança: AWS Estrutura bem arquitetada.

Use as chamadas de API publicadas da AWS para acessar o API Gateway por meio da rede. Os clientes devem oferecer compatibilidade com:

- Transport Layer Security (TLS). Exigimos TLS 1.2 e recomendamos TLS 1.3.
- Conjuntos de criptografia com perfect forward secrecy (PFS) como DHE (Ephemeral Diffie-Hellman) ou ECDHE (Ephemeral Elliptic Curve Diffie-Hellman). A maioria dos sistemas modernos, como Java 7 e versões posteriores, comporta esses modos.

É possível chamar essas operações de API de qualquer local da rede, mas o API Gateway não é compatível com políticas de acesso baseadas em recursos, que podem incluir restrições com base no endereço IP de origem. Também é possível usar políticas baseadas em recursos para controlar o acesso da Amazon Virtual Private Cloud (Amazon VPC) endpoints ou de VPCs específicas. Na verdade, isso isola o acesso à rede a um determinado recurso do API Gateway apenas por meio da VPC específica dentro da rede da AWS.

## Análise de vulnerabilidades no Amazon API Gateway

A configuração e os controles de TI são uma responsabilidade compartilhada entre a AWS e você, nosso cliente. Para obter mais informações, consulte o AWS [modelo de responsabilidade compartilhada da](#).

## Melhores práticas de segurança no Amazon API Gateway

O API Gateway fornece uma série de recursos de segurança a serem considerados no desenvolvimento e na implementação das suas próprias políticas de segurança. As melhores práticas a seguir são diretrizes gerais e não representam uma solução completa de segurança. Como essas práticas recomendadas podem não ser adequadas ou suficientes para o seu ambiente, trate-as como considerações úteis em vez de prescrições.

### Implemente o privilégio de acesso mínimo

Use políticas do IAM para implementar o acesso de menor privilégio para criar, ler, atualizar ou excluir APIs do API Gateway. Para saber mais, consulte [Gerenciamento de identidade e acesso para o Amazon API Gateway](#). O API Gateway oferece várias opções para controlar o acesso às APIs criadas por você. Para saber mais, consulte [Controlar e gerenciar o acesso a APIs REST no API Gateway](#), [Controlar e gerenciar o acesso a APIs de WebSocket no API Gateway](#) e [Controlar o acesso a APIs HTTP com autorizadores JWT no API Gateway](#).

### Implementar o registro em log

Use o CloudWatch Logs ou o Amazon Data Firehose para registrar em log solicitações em suas APIs. Para saber mais, consulte [Monitorar APIs REST no API Gateway](#), [Configurar o registro em log para APIs de WebSocket no API Gateway](#) e [Configurar registro em log para APIs HTTP no API Gateway](#).

## Implementar alarmes do Amazon CloudWatch

Ao usar alarmes do CloudWatch, você observa uma única métrica durante um período especificado. Se a métrica exceder determinado limite, uma notificação será enviada para um tópico do Amazon Simple Notification Service ou para uma política do AWS Auto Scaling. Os alarmes do CloudWatch não invocam ações quando uma métrica está em um estado específico. O estado deve ter sido alterado e mantido por uma quantidade especificada de períodos. Para obter mais informações, consulte [the section called “Métricas do CloudWatch”](#).

## Habilitar o AWS CloudTrail

O CloudTrail fornece um registro de ações executadas por um usuário, uma função ou um serviço da AWS no API Gateway. Usando as informações coletadas pelo CloudTrail, é possível determinar a solicitação feita para o API Gateway, o endereço IP do qual a solicitação foi feita, quem fez a solicitação, quando ela foi feita e detalhes adicionais. Para obter mais informações, consulte [the section called “Trabalhar com o CloudTrail”](#).

## Habilitar o AWS Config

O AWS Config fornece uma visão detalhada da configuração dos recursos da AWS na conta. Você pode ver como os recursos estão relacionados, obter um histórico de alterações de configuração e ver como os relacionamentos e as configurações foram alterados ao longo do tempo. É possível usar o AWS Config para definir regras que avaliam configurações de recursos para conformidade de dados. As regras do AWS Config representam as definições de configuração ideais para os recursos do API Gateway. Se um recurso violar uma regra e for sinalizado como não compatível, o AWS Config poderá alertá-lo usando um tópico do Amazon Simple Notification Service (Amazon SNS). Para obter detalhes, consulte [the section called “Trabalhar com o AWS Config”](#).

## Use AWS Security Hub CSPM

Monitore seu uso do API Gateway em relação às práticas recomendadas de segurança com o [AWS Security Hub CSPM](#). O CSPM Security Hub usa controles de segurança para avaliar configurações de recursos e padrões de segurança que ajudam você a cumprir vários frameworks de conformidade. Para ter mais informações sobre como usar o CSPM do Security Hub para avaliar os recursos do API Gateway, consulte [Controles do Amazon API Gateway](#) no Guia do usuário do AWS Security Hub.

# Marcar recursos do API Gateway

Uma tag é um rótulo de metadados que você ou a AWS atribui a um recurso da AWS. Cada tag tem duas partes:

- Uma chave de tag (por exemplo CostCenter, Environment ou Project). Chaves de tag fazem distinção entre maiúsculas e minúsculas.
- Um campo opcional conhecido como um valor de tag (por exemplo, 111122223333 ou Production). Omitir o valor da tag é o mesmo que usar uma string vazia. Como as chaves de tag, os valores das tags diferenciam maiúsculas de minúsculas.

As tags ajudam você a fazer o seguinte:

- Controlar o acesso aos recursos de acordo com as tags atribuídas a eles. Você controla o acesso especificando chaves e valores de tags nas condições para uma política do AWS Identity and Access Management (IAM). Para obter mais informações sobre o controle de acesso baseado em tags, consulte [Controlar o acesso usando tags](#) no Guia do usuário do IAM.
- Monitorar seus custos da AWS. Você pode ativar essas tags no painel Gerenciamento de Faturamento e Custos da AWS. AWS usa tags para categorizar seus custos e entregar um relatório mensal de alocação de custos a você. Para obter mais informações, consulte [Usar tags de alocação de custos](#) no [Guia do usuário do AWS Billing](#).
- Identificar e organizar seus recursos da AWS. Muitos serviços da AWS oferecem suporte à marcação para que você possa atribuir a mesma tag a recursos de diferentes serviços para indicar que os recursos estão relacionados. Por exemplo, é possível atribuir a mesma tag a um estágio do API Gateway que você atribui a uma regra do CloudWatch Events.

Para conferir dicas sobre como usar as etiquetas, consulte o whitepaper [AWS Tagging Strategies](#).

As próximas seções contêm mais informações sobre tags para o Amazon API Gateway.

## Tópicos

- [Recursos do API Gateway que podem ser marcados](#)
- [Usar tags para controlar o acesso aos recursos da API REST do API Gateway](#)

# Recursos do API Gateway que podem ser marcados

As tags podem ser definidas nos recursos da API HTTP a seguir ou da API WebSocket na [API do Amazon API Gateway V2](#):

- Api
- DomainName
- Stage
- VpcLink

Além disso, tags podem ser definidas nos recursos a seguir da API REST na [API do Amazon API Gateway V1](#):

- ApiKey
- ClientCertificate
- DomainName
- DomainNameAccessAssociation
- RestApi
- Stage
- UsagePlan
- VpcLink

As tags não podem ser definidas diretamente em outros recursos. No entanto, na [API do Amazon API Gateway V1](#), os recursos filho herdam as tags que são definidas nos recursos pai. Por exemplo:

- Se uma etiqueta for definida em um recurso RestApi, ela será herdada pelos seguintes recursos filho da RestApi para [Controle de acesso baseado em atributos](#):
  - Authorizer
  - Deployment
  - Documentation
  - GatewayResponse
  - Integration
  - Method



- Model
  - Resource
  - ResourcePolicy
  - Setting
  - Stage
- Se uma tag for definida em um DomainName, ela será herdada por todos os recursos BasePathMapping, ApiMapping e RoutingRule existentes nela.
  - Se uma tag for definida em um UsagePlan, essa tag será herdada por todos os recursos UsagePlanKey sob ela.

#### Note

A herança de etiquetas vale apenas para o [controle de acesso baseado em atributos](#). Por exemplo, não é possível usar etiquetas herdadas para monitorar custos no AWS Cost Explorer. Para usar tags para alocação de custos, recomendamos que você crie tags em recursos secundários, como o recurso Stage.

O API Gateway não retorna etiquetas herdadas quando você chama [GetTags](#) para um recurso.

## Herança de tags na API do Amazon API Gateway V1

Anteriormente, só era possível definir tags em estágios. Agora que você também pode defini-las em outros recursos, um Stage pode receber uma tag de duas formas:

- A tag pode ser definida diretamente no Stage.
- O estágio pode herdar a tag do recurso pa RestApi.

Se um estágio receber uma tag das duas formas, a tag que foi definida diretamente no estágio terá precedência. Por exemplo, suponha que um estágio herde as seguintes tags de sua API REST pai:

```
{
 'foo': 'bar',
 'x': 'y'
}
```

Suponha que ele também tenha as seguintes tags definidas diretamente:

```
{
 'foo': 'bar2',
 'hello': 'world'
}
```

O efeito final seria que o estágio tem as seguintes tags, com os seguintes valores:

```
{
 'foo': 'bar2',
 'hello': 'world'
 'x': 'y'
}
```

## Restrições de tags e convenções de uso

As restrições e convenções a seguir se aplicam ao uso de tags com recursos do API Gateway:

- Cada recurso pode ter um máximo de 50 tags.
- Em todos os recursos, cada chave de tag deve ser exclusiva e pode ter apenas um valor.
- O comprimento máximo da chave da tag é de 128 caracteres Unicode em UTF-8.
- O comprimento máximo do valor da tag é de 256 caracteres Unicode em UTF-8.
- Os caracteres permitidos para chaves e valores são letras, números, espaços representáveis em UTF-8, além dos seguintes caracteres: . : + = @ \_ / - (hífen). Os recursos do Amazon EC2 permitem qualquer caractere.
- As chaves e os valores de tags diferenciam maiúsculas de minúsculas. Como melhor prática, adote uma estratégia para letras maiúsculas em tags e implemente-a de forma consistente em todos os tipos de recursos. Por exemplo, decida se deseja usar Costcenter, costcenter ou CostCenter e use a mesma convenção para todas as tags. Evite usar tags semelhantes com tratamento do tamanho de letra inconsistente.
- O prefixo `aws :` é proibido em tags, pois ele é reservado para uso pela AWS. Você não pode editar nem excluir chaves nem valores de tag com esse prefixo. As tags com esse prefixo não contam para as tags por limite de recurso.

# Usar tags para controlar o acesso aos recursos da API REST do API Gateway

Condições nas políticas do AWS Identity and Access Management são parte da sintaxe que você usa para especificar permissões para recursos do API Gateway. Para obter detalhes sobre como especificar políticas do IAM, consulte [the section called “Usar permissões do IAM”](#). No API Gateway, os recursos podem ter tags, e algumas ações podem incluir tags. Ao criar uma política do IAM, você poderá usar chaves de condição de tag para controlar:

- Quais usuários podem executar ações em um recurso do API Gateway, com base nas tags que o recurso já tem.
- Quais tags podem ser transmitidas na solicitação de uma ação.
- Se chaves de tags específicas podem ser usadas em uma solicitação.

O uso de etiquetas para o controle de acesso baseado em atributo pode permitir um controle mais preciso que o controle em nível de API, bem como um controle mais dinâmico que o controle de acesso baseado em recursos. As políticas do IAM podem ser criadas para permitir ou não uma operação baseada em tags fornecidas na solicitação (tags de solicitação) ou tags no recurso em que estão sendo operadas (tags de recurso). Em geral, as tags de recurso são para recursos que já existem. As tags de solicitação são para quando você estiver criando novos recursos.

Para obter a sintaxe e a semântica completas das chaves de condição de tag, consulte [Controlar o acesso usando tags](#) no Guia do usuário do IAM.

Os exemplos a seguir demonstram como especificar condições de tag em políticas para usuários do API Gateway.

## Limitar ações com base em tags de recursos

O exemplo de política a seguir concede aos usuários permissão para executar todas as ações em todos os recursos, desde que esses recursos não tenham a tag `Environment` com um 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 ações com base em tags de recursos

O exemplo de política a seguir permite que os usuários executem todas as ações nos recursos do API Gateway, desde que os recursos tenham a tag `Environment` com um valor de `Development`. A declaração `Deny` impede que o usuário altere o valor da tag `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"
 }
 }
 }
]
}

```

## Negar operações de marcação

O exemplo de política a seguir permite que um usuário execute todas as ações do API Gateway, exceto para alterar as tags.

## JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "apigateway:*"
],
 "Resource": [
 "*"
]
 },
 {
 "Effect": "Deny",
 "Action": [
 "apigateway:*"
],
 "Resource": "arn:aws:apigateway:*::/tags*"
 }
]
}
```

## Permitir operações de marcação

O exemplo de política a seguir permite que um usuário obtenha todos os recursos do API Gateway e altere as tags para esses recursos. Para obter as tags de um recurso, o usuário deve ter permissões GET para esse recurso. Para atualizar as tags de um recurso, o usuário deve ter permissões PATCH para esse 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:*::*"
]
}
]
```

# Referências de API

O Amazon API Gateway fornece APIs para criar e implantar suas próprias APIs HTTP e WebSocket. Além disso, as APIs do API Gateway estão disponíveis em SDKs da AWS padrão.

Se estiver usando uma linguagem para a qual exista um SDK da AWS, talvez prefira utilizar o SDK em vez de optar diretamente pelas APIs REST do API Gateway. Os SDKs simplificam a autenticação, integram-se facilmente ao ambiente de desenvolvimento e fornecem acesso fácil aos comandos do API Gateway.

Veja abaixo onde encontrar os SDKs da AWS e a documentação de referência da API REST do API Gateway:

- [Ferramentas para a Amazon Web Services](#)
- [Referência da API REST do Amazon API Gateway](#)
- [Referência de API WebSocket e HTTP do Amazon API Gateway](#)

## Endpoints do serviço do API Gateway

Endpoint é um URL que funciona como ponto de entrada para um serviço da Web da AWS. A API Gateway comporta os seguintes tipos de endpoint:

- [the section called “Endpoints IPv4”](#)
- [Endpoints de pilha dupla](#) (comportam IPv4 e IPv6)
- [Endpoints do FIPS](#)

Ao fazer uma solicitação, você pode especificar o endpoint a ser usado. Se você não especificar um endpoint, o endpoint IPv4 será usado por padrão. Para usar outro tipo de endpoint, você deve especificá-lo em sua solicitação. Para obter exemplos de como fazer isso, consulte [the section called “Especificar endpoints”](#). Para ver uma tabela de endpoints disponíveis, consulte [Amazon API Gateway endpoints](#).

## Endpoints IPv4

Endpoints IPv4 só são compatíveis com tráfego IPv4. Os endpoints IPv4 estão disponíveis em todas as regiões.



Se você especificar o endpoint geral, `apigateway.amazonaws.com`, usaremos o endpoint para `us-east-1`. Para usar uma região diferente, especifique o endpoint associado a ela. Por exemplo, se você especificar `apigateway.us-east-2.amazonaws.com` como endpoint, direcionaremos sua solicitação para o endpoint `us-east-2`.

Os nomes de endpoints IPv4 usam a seguinte convenção de nomenclatura:

- `apigateway.region.amazonaws.com`

Por exemplo, o nome do endpoint IPv4 para a região `eu-west-1` é `apigateway.eu-west-1.amazonaws.com`.

## Endpoints de pilha dupla (IPv4 e IPv6)

Endpoints de pilha dupla são compatíveis com tráfego IPv4 e IPv6. Quando você realiza uma solicitação para um endpoint de pilha dupla, o URL do endpoint é resolvido para um endereço IPv6 ou IPv4, dependendo do protocolo usado pela rede e pelo cliente.

Os nomes de endpoints de pilha dupla usam a seguinte convenção de nomenclatura:

- `apigateway.region.api.aws`

Por exemplo, o nome do endpoint de pilha dupla para a região `eu-west-1` é `apigateway.eu-west-1.api.aws`.

## Especificar endpoints

Os exemplos a seguir mostram como especificar um endpoint para a região `us-east-2` usando a AWS CLI para `apigateway`.

- Pilha dupla

```
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
```

Os exemplos a seguir mostram como especificar um endpoint para a região us-east-2 usando a AWS CLI para apigatewayv2.

- Pilha dupla

```
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
```

## Como usar endereços do IPv6 em políticas do IAM

Se você usa políticas de usuário do IAM ou políticas de recursos do API Gateway para controlar o acesso ao API Gateway ou a qualquer API do API Gateway, confirme se suas políticas foram atualizadas para incluir intervalos de endereços IPv6. Políticas que não são atualizadas para lidar com endereços IPv6 podem afetar o acesso do cliente ao API Gateway quando eles começarem a usar o endpoint de pilha dupla. Para ter mais informações sobre como gerenciar permissões de acesso com o IAM, consulte [Gerenciamento de identidade e acesso para o Amazon API Gateway](#).

As políticas do IAM que filtram endereços IP usam [Operadores de condição de endereço IP](#). A política de identidade a seguir permite que endereços IP no intervalo 54.240.143.\* tenham informações sobre todos os recursos de uma API HTTP ou de WebSocket com o identificador a123456789. Quaisquer endereços IP fora desse intervalo terão o acesso negado às informações sobre todos os recursos na API. Como todos os endereços IPv6 estão fora do intervalo permitido, essa política impede que os endereços do IPv6 acessem as informações sobre a 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"}
}
]
```

É possível modificar o elemento `Condition` da política de API para permitir os intervalos de endereço IPv4 (54.240.143.0/24) e IPv6 (2001:DB8:1234:5678::/64), conforme mostrado no exemplo a seguir. É possível usar o mesmo tipo de bloqueio de `Condition` mostrado no exemplo para atualizar as políticas de usuário do IAM e de recursos do API Gateway.

```
"Condition": {
 "IpAddress": {
 "aws:SourceIp": [
 "54.240.143.0/24",
 "2001:DB8:1234:5678::/64"
]
 }
}
```

# Cotas do Amazon API Gateway

As cotas a seguir se aplicam a todos os tipos de API do Amazon API Gateway.

## Cotas no nível da conta do API Gateway, por região

As cotas a seguir são aplicáveis por conta, por região no Amazon API Gateway.

| Recurso ou operação                                                                                                                                                                                                                                                                                                                     | Cota padrão                                                                                                                                                                                                        | Pode ser aumentado  |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------|
| Controlar a cota por conta, por região em APIs HTTP, APIs REST, APIs do WebSocket e APIs de retorno de chamada do WebSocket                                                                                                                                                                                                             | 10.000 solicitações por segundo (RPS) com uma capacidade de intermitência adicional fornecida pelo <a href="#">algoritmo de bucket de tokens</a> , usando uma capacidade máxima de bucket de 5.000 solicitações. * | <a href="#">Sim</a> |
| <div>  <b>Note</b><br/>           A cota de intermitência é determinada pela equipe de serviço do API Gateway com base na cota de RPS geral da conta na região. Não é uma cota que o cliente possa controlar ou solicitar alterações.         </div> |                                                                                                                                                                                                                    |                     |
| Cota de controle de utilização sem controle de acesso por conta por região para um portal                                                                                                                                                                                                                                               | 250,000 solicitações por segundo                                                                                                                                                                                   | Não                 |
| Cota de controle de utilização com controle de acesso por conta por                                                                                                                                                                                                                                                                     | 10,000 solicitações por segundo                                                                                                                                                                                    | Não                 |

| Recurso ou operação | Cota padrão | Pode ser aumentado |
|---------------------|-------------|--------------------|
|---------------------|-------------|--------------------|

região para um portal

\* Para as seguintes regiões, a cota de controle de utilização padrão é 2.500 RPS e a cota de expansão padrão é 1.250 RPS: África (Cidade do Cabo), Europa (Milão), Ásia-Pacífico (Jacarta), Oriente Médio (EAU), Ásia-Pacífico (Hyderabad), Ásia-Pacífico (Melbourne), Europa (Espanha), Europa (Zurique), Israel (Tel Aviv), Oeste do Canadá (Calgary), Ásia-Pacífico (Malásia), Ásia-Pacífico (Tailândia) e México (Centro).

## Cotas do API Gateway para criação, implantação e gerenciamento de uma API

As cotas fixas a seguir se aplicam à criação, à implantação e ao gerenciamento de uma API no API Gateway, via AWS CLI, console do API Gateway ou API REST do API Gateway e seus SDKs. Essas cotas não podem ser aumentadas.

| Ação                                       | Cota padrão                                | Pode ser aumentado |
|--------------------------------------------|--------------------------------------------|--------------------|
| <a href="#">CreateApiKey</a>               | 5 solicitações por segundo por conta       | Não                |
| <a href="#">CreateDeployment</a>           | 1 solicitação a cada 5 segundos por conta  | Não                |
| <a href="#">CreateDocumentationVersion</a> | 1 solicitação a cada 20 segundos por conta | Não                |
| <a href="#">CreateDomainName</a>           | 1 solicitação a cada 30 segundos por conta | Não                |
| <a href="#">CreateResource</a>             | 5 solicitações por segundo por conta       | Não                |

| Ação                                                         | Cota padrão                                | Pode ser aumentado |
|--------------------------------------------------------------|--------------------------------------------|--------------------|
| <a href="#">CreateRestApi</a> para API regional ou privada.  | 1 solicitação a cada 3 segundos por conta  | Não                |
| <a href="#">CreateRestApi</a> para API otimizada para borda. | 1 solicitação a cada 30 segundos por conta | Não                |
| <a href="#">CreateVpcLink</a> (V2)                           | 1 solicitação a cada 15 segundos por conta | Não                |
| <a href="#">DeleteApiKey</a>                                 | 5 solicitações por segundo por conta       | Não                |
| <a href="#">DeleteDomainName</a>                             | 1 solicitação a cada 30 segundos por conta | Não                |
| <a href="#">DeleteResource</a>                               | 5 solicitações por segundo por conta       | Não                |
| <a href="#">DeleteRestApi</a>                                | 1 solicitação a cada 30 segundos por conta | Não                |
| <a href="#">GetResources</a>                                 | 5 solicitações a cada 2 segundos por conta | Não                |
| <a href="#">DeleteVpcLink</a> (V2)                           | 1 solicitação a cada 30 segundos por conta | Não                |
| <a href="#">ImportDocumentationParts</a>                     | 1 solicitação a cada 30 segundos por conta | Não                |
| <a href="#">ImportRestApi</a> para API regional ou privada.  | 1 solicitação a cada 3 segundos por conta  | Não                |
| <a href="#">ImportRestApi</a> para API otimizada para borda. | 1 solicitação a cada 30 segundos por conta | Não                |
| <a href="#">PutRestApi</a>                                   | 1 solicitação por segundo por conta        | Não                |

| Ação                             | Cota padrão                                | Pode ser aumentado |
|----------------------------------|--------------------------------------------|--------------------|
| <a href="#">UpdateAccount</a>    | 1 solicitação a cada 20 segundos por conta | Não                |
| <a href="#">UpdateDomainName</a> | 1 solicitação a cada 30 segundos por conta | Não                |
| <a href="#">UpdateUsagePlan</a>  | 1 solicitação a cada 20 segundos por conta | Não                |
| Criar portal                     | 1 pedido a cada 3 segundos                 | Não                |
| Atualizar portal                 | 2 solicitações por minuto                  | Não                |
| Receber portal                   | 10 solicitações por segundo                | Não                |
| Listar portais                   | 10 solicitações por segundo                | Não                |
| Publicar portal                  | 2 solicitações por minuto                  | Não                |
| DeletePortal                     | 2 solicitações por minuto                  | Não                |
| PreviewPortal                    | 1 pedido a cada 3 segundos                 | Não                |
| DisablePortal                    | 2 solicitações por minuto                  | Não                |
| GetPortalProduct                 | 10 solicitações por segundo                | Não                |
| ListPortalProduct                | 5 solicitações por segundo                 | Não                |
| CreatePortalProduct              | 2 solicitações por segundo                 | Não                |
| UpdatePortalProduct              | 0.5 solicitações por segundo               | Não                |
| DeletePortalProduct              | 1 pedido por segundo                       | Não                |
| PutPortalProductSharingPolicy    | 1 pedido a cada 3 segundos                 | Não                |
| GetPortalProductSharingPolicy    | 10 solicitações por segundo                | Não                |

| Ação                             | Cota padrão                                                                               | Pode ser aumentado |
|----------------------------------|-------------------------------------------------------------------------------------------|--------------------|
| DeletePortalProductSharingPolicy | 1 pedido a cada 3 segundos                                                                | Não                |
| CreateProductRestEndpointPage    | 1 pedido a cada 3 segundos                                                                | Não                |
| GetProductRestEndpointPage       | 10 solicitações por segundo                                                               | Não                |
| UpdateProductRestEndpointPage    | 1 pedido a cada 3 segundos                                                                | Não                |
| DeleteProductRestEndpointPage    | 1 pedido a cada 3 segundos                                                                | Não                |
| ListProductRestEndpointPages     | 10 solicitações por segundo                                                               | Não                |
| CreateProductPage                | 1 pedido a cada 3 segundos                                                                | Não                |
| GetProductPage                   | 10 solicitações por segundo                                                               | Não                |
| UpdateProductPage                | 1 pedido a cada 3 segundos                                                                | Não                |
| DeleteProductPage                | 1 pedido a cada 3 segundos                                                                | Não                |
| ListProductPages                 | 10 solicitações por segundo                                                               | Não                |
| Outras operações                 | Nenhuma cota até a cota total da conta.                                                   | Não                |
| Totais de operações              | 10 solicitações por segundo com uma cota de intermitência de 40 solicitações por segundo. | Não                |



## Cotas para configurar e executar uma API REST no API Gateway

As cotas a seguir se aplicam à configuração e à execução de uma API REST no Amazon API Gateway. Se a cota for por API, ela só poderá ser aumentada em APIs específicas, não em todas as APIs em uma conta.

Para ter informações sobre cotas em nível de conta, consulte [Cotas e observações importantes](#).

| Recurso ou operação                                             | Cota padrão | Pode ser aumentada  |
|-----------------------------------------------------------------|-------------|---------------------|
| Nomes de domínio personalizados públicos por conta e por região | 120         | <a href="#">Sim</a> |
| Nomes de domínio personalizados privados por conta e por região | 50          | <a href="#">Sim</a> |
| APIs regionais por região                                       | 600         | Não                 |
| APIs otimizadas para borda por região                           | 120         | Não                 |
| APIs privadas por conta e por região                            | 600         | Não                 |
| Associações de acesso a nomes de domínio por conta              | 100         | <a href="#">Sim</a> |
| Mapeamentos de API de vários níveis por domínio                 | 200         | Não                 |

| Recurso ou operação                                               | Cota padrão | Pode ser aumentada  |
|-------------------------------------------------------------------|-------------|---------------------|
| Regras de roteamento por domínio                                  | 50          | <a href="#">Sim</a> |
| Quantidade de caracteres do URL de uma API otimizada para a borda | 8192        | Não                 |
| Quantidade de caracteres do URL de uma API regional               | 10240       | Não                 |
| Quantidade de caracteres do URL de uma API regional               | 8192        | Não                 |
| Quantidade de caracteres da política de recursos do API Gateway   | 8192        | <a href="#">Sim</a> |
| Chaves de API por conta por região                                | 10000       | Não                 |
| Certificados de cliente por conta por região                      | 60          | <a href="#">Sim</a> |

| Recurso ou operação                                 | Cota padrão | Pode ser aumentada                                                                              |
|-----------------------------------------------------|-------------|-------------------------------------------------------------------------------------------------|
| Autorizadores por API (AWS Lambda e Amazon Cognito) | 10          | Sim<br><br>Para aumentar essa cota, entre em contato com o <a href="#">AWS Support Center</a> . |
| Partes da documentação por API                      | 2000        | Sim<br><br>Para aumentar essa cota, entre em contato com o <a href="#">AWS Support Center</a> . |
| Recursos por API                                    | 300         | <a href="#">Sim</a>                                                                             |
| Estágios por API                                    | 10          | <a href="#">Sim</a>                                                                             |
| Variáveis de estágio por estágio                    | 100         | Não                                                                                             |

| Recurso ou operação                                          | Cota padrão                                                                                                                                 | Pode ser aumentada                |
|--------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| Quantidade de caracteres da chave em uma variável de estágio | 64                                                                                                                                          | Não                               |
| Quantidade de caracteres do valor em uma variável de estágio | 512                                                                                                                                         | Não                               |
| Planos de uso por conta por região                           | 300                                                                                                                                         | <a href="#">Sim</a>               |
| Planos de uso por chave de API                               | 10                                                                                                                                          | <a href="#">Sim</a>               |
| Links de VPC por conta por região                            | 20                                                                                                                                          | <a href="#">Sim</a>               |
| TTL de armazenamento em cache de APIs                        | 300 segundos por padrão e configurável entre 0 e 3600 por um proprietário de API.                                                           | Não para o limite superior (3600) |
| Tamanho de resposta em cache                                 | 1048576 bytes. A criptografia de dados de cache pode aumentar o tamanho do item que está sendo armazenado em cache.                         | Não                               |
| Tempo limite de integração para APIs regionais               | 50 milissegundos a 29 segundos para todos os tipos de integração, incluindo integrações do Lambda, proxy do Lambda, HTTP, proxy HTTP e AWS. | <a href="#">Sim</a> *             |

| Recurso ou operação                                                                                                                                  | Cota padrão                                                                                                                                 | Pode ser aumentada    |
|------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|
| Tempo limite de integração para APIs otimizadas para borda                                                                                           | 50 milissegundos a 29 segundos para todos os tipos de integração, incluindo integrações do Lambda, proxy do Lambda, HTTP, proxy HTTP e AWS. | Não                   |
| Tempo limite de integração para APIs privadas                                                                                                        | 50 milissegundos a 29 segundos para todos os tipos de integração, incluindo integrações do Lambda, proxy do Lambda, HTTP, proxy HTTP e AWS. | <a href="#">Sim</a> * |
| Tamanho total combinado de todos os valores de cabeçalho, incluindo nomes de cabeçalho, valores, quaisquer terminadores de linha e espaços em branco | 10240 bytes                                                                                                                                 | Não                   |
| Tamanho total combinado de todos os valores do cabeçalho para uma API privada                                                                        | 8.000 bytes                                                                                                                                 | Não                   |
| Tamanho da carga útil                                                                                                                                | 10 MB                                                                                                                                       | Não                   |
| Tags por estágio                                                                                                                                     | 50                                                                                                                                          | Não                   |

| Recurso ou operação                                                                                       | Cota padrão                                                                                                                                                                              | Pode ser aumentada                                                                              |
|-----------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------|
| Número de iterações em um loop<br><code>#foreach ...</code><br><code>#end</code> em modelos de mapeamento | 1000                                                                                                                                                                                     | Não                                                                                             |
| Tamanho máximo do modelo de mapeamento                                                                    | 300 KB                                                                                                                                                                                   | Não                                                                                             |
| Tamanho do modelo de log de acesso                                                                        | 3 KB                                                                                                                                                                                     | Não                                                                                             |
| Comprimento do ARN do método                                                                              | 1.600 bytes. Se seu método contiver um parâmetro de caminho e um cliente usar um valor que exceda o comprimento do ARN, a API exibirá a resposta 414 <code>Request URI too long</code> . | Não                                                                                             |
| Configurações do controle de utilização em nível de método para um estágio em um plano de utilização      | 20                                                                                                                                                                                       | Sim<br><br>Para aumentar essa cota, entre em contato com o <a href="#">AWS Support Center</a> . |

| Recurso ou operação                                                                                                 | Cota padrão                                               | Pode ser aumentada |
|---------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------|--------------------|
| Tamanho do modelo por API                                                                                           | 400 KB                                                    | Não                |
| Número de certificados em um armazenamento de confiança                                                             | Mil certificados com até 1 MB de tamanho total do objeto. | Não                |
| Tempo limite de inatividade da conexão                                                                              | 310 segundos                                              | Não                |
| Tamanho máximo do arquivo de definição de API ao usar <a href="#">restapi:import</a> ou <a href="#">restapi:put</a> | 6 MB                                                      | Não                |

\* Não é possível definir o tempo limite de integração para menos de 50 milissegundos. Você pode aumentar o tempo limite de integração para mais de 29 segundos, mas isso pode exigir uma redução na cota de controle de utilização no nível de região da sua conta.

## Práticas recomendadas para reduzir cotas

As práticas recomendadas a seguir podem ajudar a reduzir o número atual de recursos para evitar o aumento da cota. Verifique se essas sugestões funcionam para a arquitetura da sua API.

### APIs por região

Para reduzir o número de APIs por região, exporte todas as APIs não utilizadas e exclua-as do API Gateway. Para obter mais informações, consulte [the section called “Exportar uma API REST”](#).

### Estágios por API

Para reduzir o número de estágios por API, divida a API em várias APIs.

## Recursos por API

Use caminhos {proxy+} para reduzir o número de recursos. Para obter mais informações, consulte [the section called “Configurar integrações de proxy com um recurso de proxy”](#).

## Mapeamentos de API

Para reduzir o número de mapeamentos de API para um nome de domínio personalizado, use mapeamentos de API com um único nível, como /prod. Os mapeamentos de API com um único nível não entram na contagem da cota de mapeamentos de API.

## Autorizadores por API

Para reduzir o número de autorizadores por API, reutilize os autorizadores em todos os métodos da API.

## Partes da documentação por API

Para reduzir o número de partes da documentação por API, use ignore=documentation ao importar a API. Para obter mais informações, consulte [the section called “Importar a documentação da API”](#).

Você também pode usar a herança de conteúdo para permitir que algumas partes da documentação herdem conteúdo de uma entidade de API com uma especificação mais geral. Para obter mais informações, consulte [the section called “Representação da documentação da API no API Gateway”](#).

## Quantidade de caracteres da política de recursos do API Gateway

Para reduzir o tamanho de uma política de recursos, use o AWS WAFV2 para proteger a API. Para obter mais informações, consulte [the section called “AWS WAF”](#).

Se a política contiver endereços IP, você também pode usar intervalos em vez de valores específicos para definir endereços IP.

## Planos de uso por chave de API

Para reduzir o número de planos de uso por chave de API, use uma chave de API por plano de uso e associe o plano de uso a várias APIs. Não recomendamos compartilhar uma chave de API em vários planos de uso.



## Cotas para configurar e executar uma API HTTP no API Gateway

As cotas a seguir são aplicáveis à configuração e à execução de uma API HTTP no Amazon API Gateway. Se a cota for por API, ela só poderá ser aumentada em APIs específicas, não em todas as APIs em uma conta. Para ter informações sobre cotas em nível de conta, consulte [Cotas e observações importantes](#).

| Recurso ou operação                                                        | Cota padrão                                                                                           | Pode ser aumentado  |
|----------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|---------------------|
| Rotas por API                                                              | 300                                                                                                   | <a href="#">Sim</a> |
| Integrações por API                                                        | 300                                                                                                   | Não                 |
| Tempo limite máximo de integração                                          | 30 segundos                                                                                           | Não                 |
| Estágios por API                                                           | 10                                                                                                    | <a href="#">Sim</a> |
| Mapeamentos de API de vários níveis por domínio                            | Duzentos mapeamentos de API. Esse limite não inclui mapeamento de API com um único nível, como /prod. | Não                 |
| Tags por estágio                                                           | 50                                                                                                    | Não                 |
| Tamanho total combinado dos valores de linha e de cabeçalho da solicitação | 10.240 bytes                                                                                          | Não                 |
| Tamanho da carga útil                                                      | 10 MB                                                                                                 | Não                 |

| Recurso ou operação                          | Cota padrão | Pode ser aumentado                                                                              |
|----------------------------------------------|-------------|-------------------------------------------------------------------------------------------------|
| Domínios personalizados por conta por região | 120         | <a href="#">Sim</a>                                                                             |
| Tamanho do modelo de log de acesso           | 3 KB        | Não                                                                                             |
| Entrada de log do Amazon CloudWatch Logs     | 1 MB        | Não                                                                                             |
| Autorizadores por API                        | 10          | Sim<br><br>Para aumentar essa cota, entre em contato com o <a href="#">AWS Support Center</a> . |
| Públicos-alvo por autorizador                | 50          | Não                                                                                             |
| Escopos por rota                             | 10          | Não                                                                                             |

| Recurso ou operação                                            | Cota padrão   | Pode ser aumentado  |
|----------------------------------------------------------------|---------------|---------------------|
| Tempo limite para o endpoint do conjunto de chaves Web JSON    | 1500 ms       | Não                 |
| Tamanho da resposta do endpoint do conjunto de chaves Web JSON | 150.000 bytes | Não                 |
| Tempo limite para o endpoint de descoberta do OpenID Connect   | 1500 ms       | Não                 |
| Tempo limite para resposta do autorizador do Lambda            | 10.000 ms     | Não                 |
| Links de VPC por conta por região                              | 10            | <a href="#">Sim</a> |
| Sub-redes por link de VPC                                      | 10            | <a href="#">Sim</a> |
| Variáveis de estágio por estágio                               | 100           | Não                 |
| Quantidade de caracteres da chave em uma variável de estágio   | 64            | Não                 |

| Recurso ou operação                                          | Cota padrão | Pode ser aumentado |
|--------------------------------------------------------------|-------------|--------------------|
| Quantidade de caracteres do valor em uma variável de estágio | 512         | Não                |

## Práticas recomendadas para reduzir cotas

As práticas recomendadas a seguir podem ajudar a reduzir o número atual de recursos para evitar o aumento da cota. Verifique se essas sugestões funcionam para a arquitetura da sua API.

### APIs por região

Para reduzir o número de APIs por região, exporte todas as APIs não utilizadas e exclua-as do API Gateway. Para obter mais informações, consulte [the section called “Export”](#).

### Estágios por API

Para reduzir o número de estágios por API, divida a API em várias APIs.

### Autorizadores por API

Para reduzir o número de autorizadores por API, reutilize os autorizadores em todos os métodos da API.

## Cotas para configuração de portais no API Gateway

As cotas a seguir se aplicam à criação de portais no API Gateway. Para obter mais informações, consulte [Portais do API Gateway](#).

| Recurso ou operação | Cota padrão | Pode ser aumentada |
|---------------------|-------------|--------------------|
| Portais por conta   | 15          | Sim                |

| Recurso ou operação                                                      | Cota padrão       | Pode ser aumentada |
|--------------------------------------------------------------------------|-------------------|--------------------|
| Produtos de portal por portal                                            | 200               | Não                |
| Produtos de portal por conta                                             | 500               | Não                |
| Páginas de endpoint REST do produto por produto do portal                | 40                | Sim                |
| Páginas de produto por produto do portal                                 | 40                | Sim                |
| Tamanho do logo                                                          | 200 KB            | Não                |
| Tamanho da página de documentação por página de endpoint REST do produto | 32,000 caracteres | Não                |
| Tamanho de página personalizado para páginas de produtos                 | 32,000 caracteres | Não                |
| Nomes de domínio personalizados por portal                               | 1                 | Não                |
| Autorizadores por portal                                                 | 1                 | Não                |

## Cotas para configurar e executar um WebSocket no API Gateway.

As cotas a seguir se aplicam à configuração e à execução de uma API de WebSocket no Amazon API Gateway. Se a cota for por API, ela só poderá ser aumentada em APIs específicas, não em todas as APIs em uma conta. Para ter informações sobre cotas em nível de conta, consulte [Cotas e observações importantes](#).

| Recurso ou operação                                                            | Cota padrão     | Pode ser aumentado                                                                              |
|--------------------------------------------------------------------------------|-----------------|-------------------------------------------------------------------------------------------------|
| Novas conexões por segundo e por conta (em todas as APIs WebSocket) por região | 500             | <a href="#">Sim</a>                                                                             |
| Conexões simultâneas                                                           | Não aplicável * | Não aplicável                                                                                   |
| AWS LambdaAuthorizers por API                                                  | 10              | Sim<br><br>Para aumentar essa cota, entre em contato com o <a href="#">AWS Support Center</a> . |
| AWS LambdaTokenSize do autorizador do                                          | 8 KB            | Não                                                                                             |

| Recurso ou operação                    | Cota padrão                                                                                                                                 | Pode ser aumentado                                                                              |
|----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------|
| Rotas por API                          | 300                                                                                                                                         | <a href="#">Sim</a>                                                                             |
| Integrações por API                    | 300                                                                                                                                         | Sim<br><br>Para aumentar essa cota, entre em contato com o <a href="#">AWS Support Center</a> . |
| Tempo limite de integração             | 50 milissegundos a 29 segundos para todos os tipos de integração, incluindo integrações do Lambda, proxy do Lambda, HTTP, proxy HTTP e AWS. | Não                                                                                             |
| Estágios por API                       | 10                                                                                                                                          | <a href="#">Sim</a>                                                                             |
| Tamanho de quadros WebSocket           | 32 KB                                                                                                                                       | Não                                                                                             |
| Tamanho da carga da mensagem           | 128 KB **                                                                                                                                   | Não                                                                                             |
| Tamanho máximo do modelo de mapeamento | 300 KB                                                                                                                                      | Não                                                                                             |

| Recurso ou operação                                        | Cota padrão | Pode ser aumentado |
|------------------------------------------------------------|-------------|--------------------|
| Duração da conexão para API de WebSocket                   | 2 horas     | Não                |
| Tempo limite de inatividade                                | 10 minutos  | Não                |
| Comprimento, em caracteres, do URL de uma API do WebSocket | 4096        | Não                |
| Tamanho do modelo de log de acesso                         | 3 KB        | Não                |

\* O API Gateway não impõe uma cota em conexões simultâneas. O número máximo de conexões simultâneas é determinado pela taxa de novas conexões por segundo e duração máxima da conexão de duas horas. Por exemplo, com a cota padrão de 500 novas conexões por segundo, se os clientes se conectarem à taxa máxima em duas horas, o API Gateway pode atender até 3.600.000 conexões simultâneas.

\*\* Devido à cota de tamanho de quadro do WebSocket de 32 KB, uma mensagem com mais de 32 KB deve ser dividida em vários quadros, cada um contendo 32 KB ou menos. Isso se aplica aos comandos `@connections`. Se uma mensagem maior (ou quadro de tamanho maior) for recebida, a conexão será encerrada com o código 1009.

## Notas importantes do Amazon API Gateway

A seção a seguir detalha as observações que podem afetar o uso do API Gateway.

### Tópicos

- [Observações importantes sobre o Amazon API Gateway para APIs HTTP](#)



- [Observações importantes do Amazon API Gateway para HTTP e APIs de WebSocket](#)
- [Notas importantes do Amazon API Gateway para APIs REST e WebSocket](#)
- [Notas importantes do Amazon API Gateway para APIs WebSocket](#)
- [Notas importantes do Amazon API Gateway para APIs REST](#)

## Observações importantes sobre o Amazon API Gateway para APIs HTTP

- As APIs HTTP convertem os cabeçalhos X-Forwarded-\* de entrada em um cabeçalho Forwarded padrão e anexam o IP de saída, o host e o protocolo.
- O API Gateway adicionará o cabeçalho Content-type à solicitação se não houver carga útil e o Content-Length for 0.

## Observações importantes do Amazon API Gateway para HTTP e APIs de WebSocket

- O Signature Version 4a não é oficialmente compatível com o Amazon API Gateway para HTTP e APIs de WebSocket.

## Notas importantes do Amazon API Gateway para APIs REST e WebSocket

- O API Gateway não é compatível com o compartilhamento de um nome de domínio personalizado em APIs REST e WebSocket.
- Nomes de estágio podem conter apenas caracteres alfanuméricos, hífen e sublinhados. O tamanho máximo é de 128 caracteres.
- Os caminhos /ping e /sping são reservados para a verificação de integridade do serviço. O uso desses recursos em nível raiz da API com domínios personalizados não conseguirá produzir o resultado esperado.
- No momento, o API Gateway limita os eventos de log a 1.024 bytes. Eventos de log maiores do que 1024 bytes, como corpos de solicitação e resposta, serão truncados pelo API Gateway antes do envio para o CloudWatch Logs.
- Atualmente, as métricas do CloudWatch limitam os nomes e valores de dimensão a 255 caracteres XML válidos. (Para obter mais informações, consulte o [Guia do usuário do CloudWatch](#).) Os valores de dimensão são uma função de nomes definidos pelo usuário, incluindo o nome da API, o

nome do rótulo (estágio) e o nome do recurso. Ao escolher esses nomes, tenha cuidado para não exceder os limites de métricas do CloudWatch.

- O tamanho máximo de um modelo de mapeamento é de 300 KB.

## Notas importantes do Amazon API Gateway para APIs WebSocket

- O API Gateway é compatível com cargas de mensagem de até 128 KB com quadros no tamanho máximo de 32 KB. Se uma mensagem exceder 32 KB, é necessário dividi-la em vários quadros, cada qual contendo 32 KB ou menos. Se uma mensagem maior for recebida, a conexão será encerrada com o código 1009.

## Notas importantes do Amazon API Gateway para APIs REST

- O caractere pipe de texto simples (|) e o caractere de chave ({ ou }) não podem ser usados em nenhuma solicitação de string de consulta de URL e devem ser codificados em URL.
- O caractere ponto-e-vírgula (;) não tem suporte para nenhuma string de consulta de URL da solicitação e resulta nos dados divididos.
- As APIs REST decodificam parâmetros de solicitação codificados em URL antes de transmiti-los para integrações de back-end. Para parâmetros de solicitação UTF-8, as APIs REST decodificam os parâmetros, depois os transmitem como unicode para integrações de back-end. As APIs REST codificam em URL o caractere de porcentagem (%) antes de passá-lo para integrações de backend.
- Ao usar o console do API Gateway para testar uma API, você pode obter a resposta “unknown endpoint errors” (erros de endpoint desconhecido) se um certificado autoassinado for apresentado ao backend, o certificado intermediário estiver ausente da cadeia de certificados ou qualquer outra exceção relacionada a certificados irreconhecíveis for lançada pelo backend.
- Para uma entidade [Resource](#) ou [Method](#) da API com integração privada, é necessário excluí-la depois de remover as referências codificadas permanentemente de um [VpcLink](#). Caso contrário, sua integração será suspensa e você receberá uma mensagem de erro informando que o link da VPC ainda está em uso, mesmo se a entidade Resource ou Method tiver sido excluída. Esse comportamento não se aplica quando a integração privada faz referência ao VpcLink por meio de uma variável de estágio.
- Os backends a seguir podem não ser compatíveis com a autenticação de cliente SSL de uma forma que seja compatível com o API Gateway:

- [NGINX](#)
- [Heroku](#)
- O API Gateway é compatível com a maior parte da [especificação do OpenAPI 2.0](#) e da [especificação do OpenAPI 3.0](#), com as seguintes exceções:
  - Segmentos de caminho só podem conter caracteres alfanuméricos, sublinhados, hifens, pontos, vírgulas, dois-pontos e chaves. Parâmetros de caminho devem ser segmentos de caminho separados. Por exemplo, "resource/{path\_parameter\_name}" é válido; "resource{path\_parameter\_name}" não é.
  - Nomes de modelo podem conter apenas caracteres alfanuméricos.
  - Para parâmetros de entrada, somente os atributos a seguir são compatíveis: name, in, required, type, description. Outros atributos são ignorados.
  - Se usado, o tipo securitySchemes deverá ser apiKey. No entanto, OAuth 2 e autenticação básica HTTP são compatíveis por meio de [autorizadores do Lambda](#); a configuração do OpenAPI é obtida por meio de [extensões do fornecedor](#).
  - O campo deprecated não tem suporte e é descartado em APIs exportadas.
  - Os modelos do API Gateway são definidos usando [esquema JSON rascunho 4](#) em vez do esquema JSON usado pelo OpenAPI.
  - O parâmetro discriminator não tem suporte em nenhum objeto de esquema.
  - Não há suporte para a tag example.
  - Não há suporte para o campo nullable.
  - exclusiveMinimum não é compatível com o API Gateway.
  - As marcas maxItems e minItems não estão incluídas na [validação de solicitação simples](#). Para resolver esse problema, atualize o modelo após a importação, antes de fazer a validação.
  - Com relação à OpenAPI 3.0, não é possível ter um oneOf, anyOf ou allOf que use \$ref para uma definição dentro do mesmo esquema. Você pode inserir diretamente o esquema ou definir um recurso de modelo distinto do API Gateway. Para obter mais informações, consulte [the section called “Criar modelos mais complexos”](#).
  - oneOf não é compatível com OpenAPI 2.0 ou a geração de SDK.
  - Não há suporte para o campo readOnly.
  - \$ref não pode ser usado para referenciar outros arquivos.
  - Definições de resposta do formulário "500": {"\$ref": "#/responses/UnexpectedError"} não têm suporte na raiz do documento do OpenAPI. Para contornar esse problema, substitua a referência pelo esquema embutido.

- Não há suporte para números do tipo Int32 ou Int64. Um exemplo é mostrado conforme a seguir:

```
"elementId": {
 "description": "Working Element Id",
 "format": "int32",
 "type": "number"
}
```

- O tipo de formato de número decimal ("format": "decimal") não tem suporte em uma definição de esquema.
- Em respostas de método, a definição de esquema deve ser de um tipo de objeto e não pode ser de tipos primitivos. Por exemplo, não há suporte para "schema": { "type": "string"}. No entanto, você pode contornar esse problema usando o seguinte tipo de objeto:

```
"schema": {
 "$ref": "#/definitions/StringResponse"
}

"definitions": {
 "StringResponse": {
 "type": "string"
 }
}
```

- O API Gateway não usa segurança no nível raiz definida na especificação do OpenAPI. Portanto, é necessário definir a segurança em um nível de operação a ser adequadamente aplicado.
- A palavra-chave default não é compatível.
- O API Gateway impõe as seguintes restrições e limitações ao lidar com métodos com a integração do Lambda ou a integração HTTP.
- Os nomes de cabeçalho e parâmetros de consulta são processados em uma forma com distinção entre letras maiúsculas e minúsculas.
- A tabela a seguir lista os cabeçalhos que podem ser descartados, remapeados ou modificados quando enviados ao seu endpoint de integração ou enviados de volta pelo seu endpoint de integração. Nesta tabela:
  - Remapped significa que o nome de cabeçalho é alterado de *<string>* para X-Amzn-Remapped-*<string>*.

Remapped Overwritten significa que o nome de cabeçalho é alterado de *<string>* para X-Amzn-Remapped-*<string>* e o valor é substituído.

| Nome do cabeçalho | Solicitação ( <b>http/http_proxy /lambda</b> ) | Resposta ( <b>http/http_proxy /lambda</b> ) |
|-------------------|------------------------------------------------|---------------------------------------------|
| Age               | Passagem                                       | Passagem                                    |
| Accept            | Passagem                                       | Descartado/<br>Passagem/<br>Passagem        |
| Accept-Charset    | Passagem                                       | Passagem                                    |
| Accept-Encoding   | Passagem                                       | Passagem                                    |
| Authorization     | Passagem *                                     | Remapeado                                   |
| Connection        | Descartado/Passagem/Dcartado                   | Remapeado                                   |
| Content-Encoding  | Passagem/Dcartado/Passagem                     | Passagem                                    |
| Content-Length    | Passagem (gerada com base no corpo)            | Passagem                                    |
| Content-MD5       | Descartado                                     | Remapeado                                   |
| Content-Type      | Passagem                                       | Passagem                                    |

| Nome do cabeçalho  | Solicitação ( <b>http/http_proxy /lambda</b> ) | Resposta ( <b>http/http_proxy /lambda</b> ) |
|--------------------|------------------------------------------------|---------------------------------------------|
| Date               | Passagem                                       | Remapeado substituído                       |
| Expect             | Descartado                                     | Descartado                                  |
| Host               | Substituído no endpoint de integração          | Descartado                                  |
| Max-Forwards       | Descartado                                     | Remapeado                                   |
| Pragma             | Passagem                                       | Passagem                                    |
| Proxy-Authenticate | Descartado                                     | Descartado                                  |
| Range              | Passagem                                       | Passagem                                    |
| Referer            | Passagem                                       | Passagem                                    |
| Server             | Descartado                                     | Remapeado substituído                       |
| TE                 | Descartado                                     | Descartado                                  |
| Transfer-Encoding  | Descartado/Descartado/Exceção                  | Descartado                                  |
| Trailer            | Descartado                                     | Descartado                                  |
| Upgrade            | Descartado                                     | Descartado                                  |

| Nome do cabeçalho | Solicitação ( <b>http/http_proxy /lambda</b> ) | Resposta ( <b>http/http_proxy /lambda</b> ) |
|-------------------|------------------------------------------------|---------------------------------------------|
| User-Agent        | Passagem                                       | Remapeado                                   |
| Via               | Descartado/Dcartado/Passagem                   | Passagem/<br>Descartado/<br>Descartado      |
| Warn              | Passagem                                       | Passagem                                    |
| WWW-Authenticate  | Descartado                                     | Remapeado                                   |

\* O cabeçalho Authorization será descartado se contiver uma assinatura do [Signature versão 4](#) ou se a autorização AWS\_IAM for usada.

- O SDK do Android de uma API gerada pelo API Gateway usa a classe `java.net.HttpURLConnection`. Essa classe lançará uma exceção sem tratamento, em dispositivos executando o Android 4.4 e anteriores, para uma resposta 401 resultante do remapeamento do cabeçalho WWW-Authenticate para X-Amzn-Remapped-WWW-Authenticate.
- Diferentemente dos SDKs Java, Android e iOS gerados pelo API Gateway de uma API, o SDK JavaScript de uma API gerada pelo API Gateway não é compatível com as novas tentativas para erros de nível 500.
- A invocação de teste de um método usa o tipo de conteúdo padrão de `application/json` e ignora especificações de todos os outros tipos de conteúdo.
- Ao enviar solicitações para uma API transmitindo o cabeçalho `X-HTTP-Method-Override`, o API Gateway substituirá o método. Portanto, para transmitir o cabeçalho ao backend, o cabeçalho precisa ser adicionado à solicitação de integração.
- Quando uma solicitação contém vários tipos de mídia em seu cabeçalho `Accept`, o API Gateway apenas respeita o primeiro tipo de mídia `Accept`. Na situação em que você não pode controlar a ordem dos tipos de mídia de `Accept` e o tipo de mídia do seu conteúdo binário não é o primeiro da lista, é possível adicionar o primeiro tipo de mídia `Accept` na lista `binaryMediaTypes` da sua

API, e o API Gateway retornará seu conteúdo como binário. Por exemplo, para enviar um arquivo JPEG usando um elemento `<img>` em um navegador, o navegador pode enviar `Accept:image/webp,image/*,*/*;q=0.8` em uma solicitação. Ao adicionar `image/webp` à lista, o endpoint `binaryMediaTypes` receberá o arquivo JPEG como binário.

- A personalização da resposta de gateway padrão para 413 `REQUEST_TOO_LARGE` não é compatível no momento.
- O API Gateway inclui um cabeçalho `Content-Type` para todas as respostas de integração. Por padrão, o tipo de conteúdo é `application/json`.



## Histórico do documento

A tabela a seguir descreve as alterações importantes na documentação desde a última versão do Amazon API Gateway. Para receber notificações sobre atualizações dessa documentação, você poderá se inscrever em um feed RSS, selecionando o botão RSS no menu superior do painel.

- Atualização mais recente da documentação: 02 de dezembro de 2025

| Alteração                                                                                                  | Descrição                                                                                                                                                                                                                                               | Data                   |
|------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|
| <a href="#">Adicionar uma API REST do API Gateway como destino para o Amazon Bedrock AgentCore Gateway</a> | Agora o API Gateway permite acionar uma API REST como destino para o Amazon Bedrock AgentCore Gateway. Para ter mais informações, consulte <a href="#">Adicionar uma API REST do API Gateway como destino para o Amazon Bedrock AgentCore Gateway</a> . | 2 de dezembro de 2025  |
| <a href="#">Integração privada ao Application Load Balancer</a>                                            | O API Gateway agora aceita integrações privadas a Application Load Balancers para APIs REST. Para acessar mais informações, consulte <a href="#">Integrações privadas para APIs REST</a> .                                                              | 21 de novembro de 2025 |
| <a href="#">Políticas de segurança para APIs REST e nomes de domínio personalizados</a>                    | O API Gateway agora aceita políticas de segurança TLS em APIs REST e nomes de domínio personalizados. Para acessar mais informações, consulte <a href="#">Políticas de segurança para APIs REST no API Gateway</a> .                                    | 19 de novembro de 2025 |

## [Streaming de resposta para APIs REST](#)

As APIs REST do API Gateway agora comportam streaming progressivo de cargas úteis de resposta para os clientes à medida que elas se tornam disponíveis. Para acessar mais informações, consulte [Transmitir a resposta de integração para suas integrações de proxy](#).

19 de novembro de 2025

## [Portais do desenvolvedor](#)

O API Gateway agora comporta portais, que são um local centralizado para seus clientes descobrirem e testarem suas APIs e produtos do portal que representam um serviço ou funcionalidade que você deseja compartilhar. Para acessar mais informações, consulte [Portais do API Gateway](#).

19 de novembro de 2025

## [SigV4a para APIs REST](#)

O API Gateway agora permite o uso do Signature Version 4a para APIs REST. Para ter mais informações, consulte [Modelo de permissões do API Gateway para invocar uma API](#).

19 de agosto de 2025

## [Regras de roteamento para APIs REST](#)

O API Gateway agora permite regras de roteamento para APIs REST usando nomes de domínio personalizados. Você pode rotear dinamicamente as solicitações recebidas com base em valores de cabeçalho HTTP, em caminhos base de URL ou em uma combinação de ambos. Para ter mais informações, consulte [Encaminhar o tráfego às APIs por meio do nome de domínio personalizado no API Gateway](#).

3 de junho de 2025

## [Adição de suporte para endpoints de pilha dupla \(IPv4 e IPv6\)](#)

O API Gateway agora comporta endpoints de pilha dupla para APIs REST, HTTP e WebSocket, além de nomes de domínio personalizados. Para ter mais informações, consulte [Tipos de endereço IP](#).

28 de março de 2025

## [Nomes de domínio personalizados para APIs privadas](#)

Suporte adicionado para nomes de domínio personalizados para APIs privadas. Para ter mais informações, consulte [Nomes de domínio personalizados para APIs privadas](#).

21 de novembro de 2024

|                                                                        |                                                                                                                                                                                                                                                                                                                                                            |                         |
|------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|
| <a href="#">Adição de suporte a TLS 1.3</a>                            | <p>O API Gateway agora oferece suporte a TLS 1.3 em APIs REST regionais, APIs HTTP e APIs de WebSocket.</p> <p>Para saber mais, consulte <a href="#">Choosing a security policy for your custom domain in API Gateway</a>.</p>                                                                                                                             | 15 de fevereiro de 2024 |
| <a href="#">Atualizações do console de API REST e API de WebSocket</a> | <p>Atualização das informações do console sobre as APIs REST e as APIs de WebSocket.</p>                                                                                                                                                                                                                                                                   | 10 de dezembro de 2023  |
| <a href="#">Atualização da documentação</a>                            | <p>Atualização das informações conceituais e criação de outros tutoriais para transformações de dados e tópicos de validação de solicitações para APIs REST do API Gateway. Para obter mais informações, consulte <a href="#">Usar a validação de solicitação no API Gateway</a> e <a href="#">Configurar transformações de dados para a API REST</a>.</p> | 22 de junho de 2023     |

### [Configurar o failover de DNS para um API Gateway de várias regiões](#)

Adição de suporte para usar as verificações de integridade de do Amazon Route 53 a fim de controlar o failover de DNS de uma API REST do API Gateway em uma Região da AWS primária para outra em uma região secundária. Para obter mais informações, consulte [Configurar verificações de integridade personalizadas para failover de DNS](#).

31 de outubro de 2022

### [Atualização da documentação](#)

Atualização de resumos dos principais atributos da API REST e das APIs da API HTTP. Para obter mais informações, consulte [Escolher entre a API REST e as APIs da API HTTP](#).

31 de maio de 2022

### [Atualização da política gerenciada](#)

Adição de suporte ao `acm:GetCertificate` para a política `AWSServiceRoleForAPIGateway`. Para obter mais informações, consulte [Usar funções vinculadas ao serviço para o API Gateway](#).

12 de julho de 2021

### [Mapeamento de parâmetros para APIs HTTP](#)

Adicionado suporte para mapeamento de parâmetros para APIs HTTP. Para obter mais informações, consulte [Transformação de solicitações e respostas de API](#).

7 de janeiro de 2021

### [Desativar o endpoint padrão para uma API REST](#)

Adição de suporte para desabilitar o endpoint padrão para APIs REST. Para obter mais informações, consulte [Desabilitar o endpoint padrão para uma API REST](#).

29 de outubro de 2020

### [Autenticação TLS mútua](#)

Adição de suporte para autenticação TLS mútua para APIs REST e APIs HTTP. Para obter mais informações, consulte [Configurar a autenticação TLS mútua para uma API REST](#) e [Configurar a autenticação TLS mútua para uma API HTTP](#).

17 de setembro de 2020

### Autorizadores do AWS Lambda da [API HTTP](#)

Adicionado suporte para autorizadores do AWS Lambda para APIs HTTP. Para obter mais informações, consulte [Trabalhando com autorizadores do AWS Lambda para APIs HTTP](#).

9 de setembro de 2020

### Integrações de serviço da AWS para [API HTTP](#)

Adicionado suporte para integrações de serviço da AWS para APIs HTTP. Para obter mais informações, consulte [Trabalhando com integrações de serviço da AWS](#) para APIs HTTP.

20 de agosto de 2020

|                                                                           |                                                                                                                                                                                                                                                          |                      |
|---------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|
| <a href="#">Domínios personalizados de caractere curinga de APIs HTTP</a> | Inclusão de suporte para nomes de domínio personalizados curinga para APIs HTTP. Para obter mais informações, consulte <a href="#">Nomes de domínio personalizados curinga</a> .                                                                         | 10 de agosto de 2020 |
| <a href="#">Melhorias no portal do desenvolvedor sem servidor</a>         | Adicionado o gerenciamento de usuários ao painel do administrador e suporte para exportar definições de API. Para obter mais informações, consulte <a href="#">Usar o portal do desenvolvedor sem servidor para catalogar suas APIs do API Gateway</a> . | 25 de junho de 2020  |
| Suporte à <a href="#">API de WebSocket Sec-WebSocket-Protocol</a>         | Adicionado suporte para o campo Sec-WebSocket-Protocol . Para obter mais informações, consulte <a href="#">Configurar uma rota \$connect que requer um subprotocolo WebSocket</a> .                                                                      | 16 de junho de 2020  |
| <a href="#">Exportação de APIs HTTP</a>                                   | Inclusão de suporte para exportar definições do OpenAPI 3.0 de APIs HTTP. Para obter mais informações, consulte <a href="#">Exportar uma API HTTP do API Gateway</a> .                                                                                   | 20 de abril de 2020  |

|                                                             |                                                                                                                                                                                              |                         |
|-------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|
| <a href="#">Documentação de segurança</a>                   | Adicionada documentação de segurança. Para obter mais informações, consulte <a href="#">Segurança no Amazon API Gateway</a> .                                                                | 31 de março de 2020     |
| <a href="#">Documentação reorganizada</a>                   | Reorganização do guia do desenvolvedor.                                                                                                                                                      | 12 de março de 2020     |
| <a href="#">Disponibilidade geral da API HTTP</a>           | APIs HTTP lançadas em disponibilidade geral. Para obter mais informações, consulte <a href="#">Trabalhar com APIs HTTP</a> .                                                                 | 12 de março de 2020     |
| <a href="#">Registro em log de API HTTP</a>                 | Adição de suporte para <code>\$context.integrationErrorMessage</code> em logs de API HTTP. Para obter mais informações, consulte <a href="#">Variáveis de registro em log de APIs HTTP</a> . | 26 de fevereiro de 2020 |
| <a href="#">AWS de variáveis para importação de OpenAPI</a> | Adicionado suporte para variáveis da AWS em definições de OpenAPI. Para obter mais informações, consulte <a href="#">Variáveis da AWS para importação de OpenAPI</a> .                       | 17 de fevereiro de 2020 |
| <a href="#">APIs HTTP</a>                                   | APIs HTTP lançadas em beta. Para obter mais informações, consulte <a href="#">APIs HTTP</a> .                                                                                                | 4 de dezembro de 2019   |



|                                                                           |                                                                                                                                                                                                                        |                        |
|---------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|
| <a href="#">Nomes de domínio personalizados curinga</a>                   | Adição de suporte para nomes de domínio personalizados curinga. Para obter mais informações, consulte <a href="#">Nomes de domínio personalizados curinga</a> .                                                        | 21 de outubro de 2019  |
| <a href="#">Registro em log do Amazon Data Firehose</a>                   | Adição de suporte ao Amazon Data Firehose como destino para dados de registro em log de acesso. Para saber mais, consulte <a href="#">Using Amazon Data Firehose as a Destination for API Gateway Access Logging</a> . | 15 de outubro de 2019  |
| <a href="#">Aliases do Route53 para invocar APIs privadas</a>             | Adição de suporte para registros DNS de alias do Route53 adicionais para invocar APIs privadas. Para obter mais informações, consulte <a href="#">Acessar sua API privada usando um alias do Route53</a> .             | 18 de setembro de 2019 |
| <a href="#">Controle de acesso baseado em tags para APIs do WebSocket</a> | Adição de suporte ao controle de acesso baseado em tags para APIs do WebSocket. Para obter mais informações, consulte <a href="#">Recursos do API Gateway que podem ser marcados</a> .                                 | 27 de junho de 2019    |

### [Seleção de versão do TLS para domínios personalizados](#)

Adição de suporte à seleção de versão do Transport Layer Security (TLS) para APIs que são implantadas em domínios personalizados. Consulte a observação em [Escolher uma versão mínima do TLS para um domínio personalizado no API Gateway](#).

20 de junho de 2019

### [Políticas de VPC endpoint para APIs privadas](#)

Adição de suporte para melhorar a segurança de APIs privadas anexando políticas de endpoint a VPC endpoints de interface. Para obter mais informações, consulte [Usar políticas de VPC endpoint para APIs privadas no API Gateway](#).

4 de junho de 2019

### [Documentação atualizada](#)

Reformulação de [Conceitos básicos do Amazon API Gateway](#). Transferência dos tutoriais para [Tutoriais do Amazon API Gateway](#).

29 de maio de 2019

### [Controle de acesso baseado em tags para APIs REST](#)

Adição de suporte ao controle de acesso baseado em tags para APIs REST. Para obter mais informações, consulte [Usar tags com políticas do IAM para controlar o acesso aos recursos do API Gateway](#).

23 de maio de 2019

|                                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                     |
|-------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------|
| <a href="#">Documentação atualizada</a>                           | Reformulação de seis tópicos:<br><a href="#">O que é o Amazon API Gateway?</a> , <a href="#">Tutorial: Criar uma API com integração de proxy HTTP</a> , <a href="#">Tutorial: Criar uma API REST Calc com três integrações não proxy</a> , <a href="#">Referência de variáveis de registro em log de acesso e modelo de mapeamento do API Gateway</a> , <a href="#">Usar autorizadores do Lambda do API Gateway</a> e <a href="#">Habilitar o CORS para um recurso da API REST do API Gateway</a> . | 5 de abril de 2019  |
| <a href="#">Melhorias no portal do desenvolvedor sem servidor</a> | Inclusão do painel do administrador e outras melhorias para facilitar a publicação de APIs no portal do desenvolvedor do Amazon API Gateway. Para obter mais informações, consulte <a href="#">Usar um portal do desenvolvedor para catalogar as APIs</a> .                                                                                                                                                                                                                                         | 28 de março de 2019 |
| <a href="#">Suporte ao AWS Config</a>                             | Adicionado o suporte para AWS Config. Para obter mais informações, consulte <a href="#">Monitorando a configuração da API do API Gateway com AWS Config</a> .                                                                                                                                                                                                                                                                                                                                       | 20 de março de 2019 |

|                                                                                                               |                                                                                                                                                                                                                                                                                                                           |                        |
|---------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|
| <a href="#">Suporte ao CloudFormation</a>                                                                     | Adicionada a API do API Gateway V2 à referência do modelo do CloudFormation. Para obter mais informações, consulte <a href="#">Referência de tipos de recursos do Amazon API Gateway V2</a> .                                                                                                                             | 7 de fevereiro de 2019 |
| <a href="#">Suporte para APIs WebSocket</a>                                                                   | Suporte adicionado para APIs WebSocket. Para obter mais informações, consulte <a href="#">Criar uma API de WebSocket no Amazon API Gateway</a> .                                                                                                                                                                          | 18 de dezembro de 2018 |
| <a href="#">Portal para desenvolvedores sem servidor disponível via AWS Serverless Application Repository</a> | A aplicação sem servidor do portal do desenvolvedor do Amazon API Gateway agora está disponível no <a href="#">AWS Serverless Application Repository</a> (além do <a href="#">GitHub</a> ). Para obter mais informações, consulte <a href="#">Usar um portal do desenvolvedor para catalogar as APIs do API Gateway</a> . | 16 de novembro de 2018 |
| <a href="#">Suporte ao AWS WAF</a>                                                                            | Suporte adicionado para <a href="#">AWS WAF</a> (Firewall de aplicativos web). Para obter mais informações, consulte <a href="#">Controlar o acesso a uma API usando o AWS WAF</a> .                                                                                                                                      | 5 de novembro de 2018  |

[Portal do desenvolvedor sem servidor](#)

Agora o Amazon API Gateway fornece um portal do desenvolvedor totalmente personalizável como uma aplicação sem servidor que pode ser implantado para publicar as APIs do API Gateway. Para obter mais informações, consulte [Usar um portal do desenvolvedor para catalogar as APIs do API Gateway](#).

29 de outubro de 2018

[Suporte para cabeçalhos de vários valores e parâmetros de string de consulta](#)

Agora o Amazon API Gateway é compatível com vários cabeçalhos e parâmetros de string de consulta que têm o mesmo nome. Para obter mais informações, consulte [Suporte para cabeçalhos de vários valores e parâmetros de string de consulta](#).

4 de outubro de 2018

[Suporte ao OpenAPI](#)

O Amazon API Gateway agora é compatível com o OpenAPI 3.0, bem como com o OpenAPI (Swagger) 2.0.

27 de setembro de 2018

[Documentação atualizada](#)

Inclusão de um novo tópico: [Como as políticas de recursos do Amazon API Gateway afetam o fluxo de trabalho de autorização](#)

27 de setembro de 2018

## [Integração do AWS X-Ray ativa](#)

Agora você pode usar o AWS X-Ray para rastrear e analisar latências em solicitações do usuário à medida que passam por suas APIs a caminho dos serviços subjacentes. Para obter mais informações, consulte [Rastrear a execução da API do API Gateway com o AWS X-Ray](#).

6 de setembro de 2018

## [Melhorias no armazenamento em cache](#)

Somente métodos GET terão o armazenamento em cache habilitado por padrão ao habilitar o armazenamento em cache para um estágio da API. Isso ajuda a garantir a segurança da sua API. Você pode habilitar o armazenamento em cache para outros métodos, substituindo as configurações do método. Para obter mais informações, consulte [Habilitar o armazenamento em cache de APIs para melhorar a capacidade de resposta](#).

20 de agosto de 2018

### Limites de serviço revistos

Vários limites foram revistos: maior número de APIs por conta. Maiores limites de taxa de API para criar/importar/implantar APIs. Algumas taxas por minuto foram corrigidas para taxas por segundo. Para obter mais informações, consulte [Limites](#).

13 de julho de 2018

### Substituição de parâmetros e cabeçalhos de solicitação e resposta de API

Suporte adicionado para substituir cabeçalhos de solicitação, strings de consulta e caminhos, bem como cabeçalhos de resposta e códigos de status. Para obter mais informações, consulte [Usar um modelo de mapeamento para substituir os parâmetros e os cabeçalhos de solicitação e resposta de uma API](#).

12 de julho de 2018

### [Limitação de uso em nível de método para planos de uso](#)

Suporte adicionado para configuração de limitações de uso por método, bem como para configuração de limitações de uso para métodos de API específicos nas configurações de plano de uso. Essas configurações são complementares à limitação de uso da conta existente e às limitações de uso em nível de método, e você pode defini-las nas configurações de estágio. Para obter mais informações, consulte [Limitar as solicitações de API para uma melhor taxa de transferência](#).

11 de julho de 2018

### [Notificações de atualização do Guia do desenvolvedor do API Gateway agora disponíveis pelo RSS](#)

A versão HTML do Guia do desenvolvedor do API Gateway agora é compatível com um feed RSS de atualizações que estão documentadas na página Histórico de documentação. O feed RSS inclui atualizações feitas em 27 de junho de 2018, e posterior. Atualizações anteriormente anunciadas ainda estão disponíveis nesta página. Use o botão RSS no menu superior do painel para assinar o arquivo.

27 de junho de 2018



## Atualizações anteriores

A tabela a seguir descreve as alterações importantes em cada versão do Guia do desenvolvedor do API Gateway antes de 27 de junho de 2018.

| Alteração                                                                                                 | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Alterado em            |
|-----------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|
| APIs privadas                                                                                             | Inclusão de suporte para <a href="#">APIs privadas</a> , que são expostas por meio de <a href="#">VPC endpoints de interface</a> . O tráfego para as APIs privadas não deixa a rede da Amazon; ele é isolado da Internet pública.                                                                                                                                                                                                                                                                                                                                                                                                                                          | 14 de junho de 2018    |
| Autorizadores e integrações do Lambda entre contas e autorizadores do grupo de usuários do Amazon Cognito | Use uma função AWS Lambda de uma conta diferente da AWS como uma função de autorizador do Lambda ou como um backend de integração de API. Ou use um grupo de usuários do Amazon Cognito como um autorizador. A outra conta pode estar em qualquer região onde o Amazon API Gateway esteja disponível. Para obter mais informações, consulte <a href="#">the section called “Configurar um autorizador do Lambda entre contas”</a> , <a href="#">the section called “Tutorial: Crie uma API REST com uma integração de proxy do Lambda entre contas”</a> e <a href="#">the section called “Configurar o autorizador do Amazon Cognito entre contas para uma API REST”</a> . | 2 de abril de 2018     |
| Políticas de recursos para APIs                                                                           | Use as políticas de recursos do API Gateway para permitir que os usuários de uma conta da AWSdiferente acessem sua API com segurança ou para permitir que a API seja invocada somente de intervalos de endereços IP de origem ou blocos CIDR especificados. Para obter mais informações, consulte <a href="#">the section called “Usar políticas de recursos do API Gateway”</a> .                                                                                                                                                                                                                                                                                         | 2 de abril de 2018     |
| Uso de tags recursos de API Gateway                                                                       | Marque um estágio de API com até 50 tags para alocação de custos de solicitações da API e armazenamento em cache no API Gateway. Para obter mais informações, consulte <a href="#">the section called “Configurar tags”</a> .                                                                                                                                                                                                                                                                                                                                                                                                                                              | 19 de dezembro de 2017 |

| Alteração                                              | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | Alterado em            |
|--------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|
| Compactação e descompactação de carga                  | Permita chamar uma API com cargas compactadas usando uma das codificações de conteúdo compatíveis. As cargas compactadas estão sujeitas ao mapeamento se um modelo de mapeamento do corpo é especificado. Para obter mais informações, consulte <a href="#">the section called “Codificação de conteúdo”</a> .                                                                                                                                                                 | 19 de dezembro de 2017 |
| Chave de API originada de um autorizador personalizado | Retorne uma chave de API a partir de um autorizador personalizado para o API Gateway aplicar um plano de uso para os métodos de API que exigem a chave. Para obter mais informações, consulte <a href="#">the section called “Escolher uma origem de chaves de API no API Gateway”</a> .                                                                                                                                                                                       | 19 de dezembro de 2017 |
| Autorização com escopos do OAuth 2                     | Permita a autorização da invocação do método usando escopos do OAuth 2 com o autorizador COGNITO_USER_POOLS . Para obter mais informações, consulte <a href="#">the section called “Use o grupo de usuários do Amazon Cognito como um autorizador para uma API REST”</a> .                                                                                                                                                                                                     | 14 de dezembro de 2017 |
| Integração e link de VPC privados                      | Crie uma API com a integração privada do API Gateway para oferecer aos clientes acesso a recursos de HTTP/HTTPS em uma Amazon VPC de fora da VPC por meio de um recurso de <a href="#">VpcLink</a> . Para obter mais informações, consulte <a href="#">the section called “Tutorial: Crie uma API REST com uma integração privada”</a> e <a href="#">the section called “Configurar uma integração privada”</a> .                                                              | 30 de novembro de 2017 |
| Implantar um canary para testes da API                 | Adicione uma versão canary a uma implantação de API existente para testar uma versão mais recente da API, mantendo a versão atual em operação no mesmo estágio. É possível definir uma porcentagem de tráfego do estágio para a versão do canário e permitir a execução específica do canário e o acesso conectado em logs do CloudWatch Logs separados. Para obter mais informações, consulte <a href="#">the section called “Definir uma implantação da versão canary”</a> . | 28 de novembro de 2017 |

| Alteração                                | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Alterado em            |
|------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|
| Registro de acesso em logs               | Registre o acesso do cliente à sua API em logs com dados derivados de <a href="#">variáveis \$context</a> em um formato de sua escolha. Para obter mais informações, consulte <a href="#">the section called “Logs do CloudWatch”</a> .                                                                                                                                                                                                                                                                                                         | 21 de novembro de 2017 |
| SDK Ruby de uma API                      | Gere um SDK Ruby para sua API e use-o para invocar métodos da API. Para obter mais informações, consulte <a href="#">the section called “Gerar o SDK Ruby de uma API no API Gateway”</a> e <a href="#">the section called “Usar um SDK Ruby gerado pelo API Gateway para uma API REST”</a> .                                                                                                                                                                                                                                                    | 20 de novembro de 2017 |
| Endpoint de API regional                 | Especifique um endpoint de API regional para criar uma API para clientes não móveis. Um cliente não móvel, como uma instância do EC2, é executado na mesma região da AWS onde a API está implantada. Assim como ocorre com uma API otimizada para fronteiras, você pode criar um nome de domínio personalizado para uma API regional. Para obter mais informações, consulte <a href="#">the section called “Tipos de endpoint do API Gateway”</a> e <a href="#">the section called “Configurar um nome de domínio regional personalizado”</a> . | 2 de novembro de 2017  |
| autorizador de solicitação personalizado | Use o autorizador de solicitação personalizado para fornecer informações de autenticação de usuário em parâmetros de solicitação para autorizar chamadas de método de API. Os parâmetros de solicitação incluem cabeçalhos e parâmetros de string de consulta, bem como variáveis de estágio e contexto. Para obter mais informações, consulte <a href="#">Usar os autorizadores do API Gateway Lambda</a> .                                                                                                                                    | 15 de setembro de 2017 |

| Alteração                                                                                   | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Alterado em         |
|---------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------|
| Como personalizar respostas de gateway                                                      | Personalize respostas de gateway geradas pelo API Gateway a solicitações de API que não conseguiram atingir o backend de integração. Uma mensagem de gateway personalizada pode fornecer ao agente de chamada mensagens de erro personalizadas específicas de API, incluindo o retorno de cabeçalhos CORS necessários, ou pode transformar os dados de resposta de gateway em um formato de intercâmbio externo. Para obter mais informações, consulte <a href="#">Configurar respostas do gateway para personalizar respostas de erro</a> . | 6 de junho de 2017  |
| Mapear propriedades de erros personalizados do Lambda para cabeçalhos de resposta de método | Mapeie propriedades individuais de erros personalizados retornadas do Lambda para os parâmetros de cabeçalho de resposta de método usando o parâmetro <code>integration.response.body</code> , baseando-se no API Gateway para desserializar o objeto de erro personalizado transformado em string em tempo de execução. Para obter mais informações, consulte <a href="#">Manipule erros personalizados do Lambda no API Gateway</a> .                                                                                                      | 6 de junho de 2017  |
| Aumento nos limites de controle de fluxo                                                    | Aumente o limite de taxas de solicitação de estado estacionário em nível de conta para 10.000 solicitações por segundo (rps) e o limite de queda para 5000 solicitações simultâneas. Para obter mais informações, consulte <a href="#">Controle de utilização das solicitações das APIs REST para ter um melhor throughput no API Gateway</a> .                                                                                                                                                                                              | 6 de junho de 2017  |
| Validando solicitações de método                                                            | Configure validadores de solicitação básicos no nível da API ou nos níveis de métodos para que o API Gateway possa validar solicitações de entrada. O API Gateway verifica se os parâmetros necessários estão definidos e não estão em branco e verifica se o formato das cargas aplicáveis está em conformidade com o modelo configurado. Para obter mais informações, consulte <a href="#">Solicitar validação para APIs REST no API Gateway</a> .                                                                                         | 11 de abril de 2017 |

| Alteração                                           | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Alterado em           |
|-----------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|
| Integrar com o ACM                                  | Use certificados do ACM para os nomes de domínio personalizados da sua API. Você pode criar um certificado no AWS Certificate Manager ou importar um certificado existente no formato PEM para o ACM. Em seguida, faça referência ao ARN do certificado ao definir um nome de domínio personalizado para as suas APIs. Para obter mais informações, consulte <a href="#">Nome de domínio personalizado para APIs REST públicas no API Gateway</a> .                                                 | 9 de março de 2017    |
| Gerando e chamando um SDK Java de uma API           | Permita que o API Gateway gere o SDK Java para a sua API e use esse SDK para chamar a API no seu cliente Java. Para obter mais informações, consulte <a href="#">Usar um SDK Java gerado pelo API Gateway para uma API REST</a> .                                                                                                                                                                                                                                                                   | 13 de janeiro de 2017 |
| Integração com o AWS Marketplace                    | Venda sua API em um plano de uso como um produto SaaS no AWS Marketplace. Use o AWS Marketplace para estender o alcance da sua API. Conte com o AWS Marketplace para a cobrança dos clientes em seu nome. Deixe o API Gateway lidar com a autorização do usuário e a medição de uso. Para obter mais informações, consulte <a href="#">Vender suas APIs como SaaS</a> .                                                                                                                             | 1 de dezembro de 2016 |
| Habilitando o suporte a documentação para a sua API | Adicione a documentação para entidades de API a recursos <a href="#">DocumentationPart</a> no API Gateway. Associe um snapshot das instâncias de <code>DocumentationPart</code> da coleção com um estágio de API para criar uma <a href="#">DocumentationVersion</a> . Publique a documentação da API exportando uma versão da documentação para um arquivo externo, como um arquivo do Swagger. Para obter mais informações, consulte <a href="#">Documentação para APIs REST no API Gateway</a> . | 1 de dezembro de 2016 |

| Alteração                                                                                                                      | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Alterado em            |
|--------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|
| Autorizador personalizado atualizado                                                                                           | Agora, uma função do Lambda do autorizador personalizado retorna o identificador principal do autor da chamada. A função também pode retornar outras informações como pares de chave/valor do mapa context e uma política do IAM. Para obter mais informações, consulte <a href="#">Saída de um autorizador do Lambda para o API Gateway</a> .                                                                                                                                                                                             | 1 de dezembro de 2016  |
| Oferecendo suporte a cargas binárias                                                                                           | Defina <a href="#">binaryMediaTypes</a> na sua API para oferecer suporte a cargas binárias de uma solicitação ou resposta. Defina a propriedade contentHandling em um recurso <a href="#">Integrati on</a> ou <a href="#">IntegrationResponse</a> para especificar se você deseja lidar com uma carga binária como o blob binário nativo, como uma string codificada em Base64 ou como uma passagem direta sem modificações. Para obter mais informações, consulte <a href="#">Tipos de mídia binários para APIs REST no API Gateway</a> . | 17 de novembro de 2016 |
| Habilitar uma integração de proxy com um backend HTTP ou Lambda por meio de um recurso de proxy de uma API.                    | Crie um recurso de proxy com parâmetro de caminho voraz no formato {proxy+} e o método genérico ANY. O recurso de proxy é integrado a um backend HTTP ou Lambda usando a integração de proxy HTTP ou Lambda, respectivamente. Para obter mais informações, consulte <a href="#">Configurar a integração de proxy com um recurso de proxy</a> .                                                                                                                                                                                             | 20 de setembro de 2016 |
| Estender APIs selecionadas no API Gateway como ofertas de produtos para os seus clientes, fornecendo um ou mais planos de uso. | Crie um plano de uso no API Gateway para permitir que clientes de API selecionados acessem estágios de API específicos a taxas e cotas de solicitação estabelecidas. Para obter mais informações, consulte <a href="#">Usar planos e chaves de API para APIs REST no APIs Gateway</a> .                                                                                                                                                                                                                                                    | 11 de agosto de 2016   |

| Alteração                                                                                                                   | Descrição                                                                                                                                                                                                                                                                                                                                                                                               | Alterado em         |
|-----------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------|
| Habilitar a autorização em nível de método com um grupo de usuários no Amazon Cognito                                       | Crie um grupo de usuários no Amazon Cognito e use-o como seu próprio provedor de identidades. Você pode configurar o grupo de usuários como um autorizador em nível de método para conceder acesso para usuários que estão registrados nesse grupo. Para obter mais informações, consulte <a href="#">Controlar o acesso a APIs REST usando grupos de usuários do Amazon Cognito como autorizador</a> . | 28 de julho de 2016 |
| Habilitando métricas e dimensões do Amazon CloudWatch no namespace AWS/ApiGateway                                           | As métricas do API Gateway agora são padronizadas sob o namespace CloudWatch do AWS/ApiGateway. É possível visualizá-los no console do API Gateway e no console do Amazon CloudWatch. Para obter mais informações, consulte <a href="#">Dimensões e métricas do Amazon API Gateway</a> .                                                                                                                | 28 de julho de 2016 |
| Habilitando o revezamento de certificados para um nome de domínio personalizado                                             | O revezamento de certificados permite que você carregue e renove um certificado prestes a expirar para um nome de domínio personalizado. Para obter mais informações, consulte <a href="#">Alternar um certificado importado para o ACM</a> .                                                                                                                                                           | 27 de abril de 2016 |
| Documentando alterações para o console do Amazon API Gateway atualizado.                                                    | Saiba como criar e configurar uma API usando o console do API Gateway atualizado. Para obter mais informações, consulte <a href="#">Tutorial: Criar uma API REST importando um exemplo</a> e <a href="#">Tutorial: Crie uma API REST com uma integração de não proxy de HTTP</a> .                                                                                                                      | 5 de abril de 2016  |
| Habilitando o recurso Import API para criar uma API nova ou atualizar uma existente a partir de definições de API externas. | Com os recursos Import API, é possível criar uma nova API ou atualizar uma existente, fazendo upload de uma definição de API externa expressa no Swagger 2.0 com as extensões do API Gateway. Para obter mais informações sobre o recurso Import API, consulte <a href="#">Desenvolver APIs REST usando OpenAPI no API Gateway</a> .                                                                    | 5 de abril de 2016  |

| Alteração                                                                                                                                                                                                         | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Alterado em             |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|
| Expondo a variável <code>\$input.body</code> para acessar a carga bruta como uma string e a função <code>\$util.parseJson()</code> para transformar uma string JSON em um objeto JSON em um modelo de mapeamento. | Para obter mais informações sobre <code>\$input.body</code> e <code>\$util.parseJson()</code> , consulte <a href="#">Variáveis para transformações de dados para o API Gateway</a> .                                                                                                                                                                                                                                                                                                                                 | 5 de abril de 2016      |
| Habilitando solicitações de cliente com a invalidação do cache em nível de método e melhorando o gerenciamento do controle de fluxo de solicitações.                                                              | Libere o cache em nível de estágio de API e invalide a entrada de cache individual. Para obter mais informações, consulte <a href="#">Liberar o cache de estágio de APIs no API Gateway</a> e <a href="#">Invalidar uma entrada de cache do API Gateway</a> . Melhore a experiência do console para gerenciar o controle de fluxo de solicitações de API. Para obter mais informações, consulte <a href="#">Controle de utilização das solicitações das APIs REST para ter um melhor throughput no API Gateway</a> . | 25 de março de 2016     |
| Habilitar e chamar a API do API Gateway com o uso de uma autorização personalizada                                                                                                                                | Crie e configure uma função AWS Lambda para implementar a autorização personalizada. A função retorna um documento de política do IAM que concede as permissões Permitir ou Negar às solicitações de clientes de uma API do API Gateway. Para obter mais informações, consulte <a href="#">Usar os autorizadores do API Gateway Lambda</a> .                                                                                                                                                                         | 11 de fevereiro de 2016 |



| Alteração                                                                                                             | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Alterado em            |
|-----------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|
| Importar e exportar a API do API Gateway com o uso de extensões e de um arquivo de definição do Swagger               | Crie e atualize sua API do API Gateway usando a especificação do Swagger com as extensões do API Gateway. Importe as definições do Swagger usando o recurso Importer do API Gateway. Exporte uma API do API Gateway para um arquivo de definição do Swagger usando o console do API Gateway ou a API de exportação do API Gateway. Para obter mais informações, consulte <a href="#">Desenvolver APIs REST usando OpenAPI no API Gateway</a> e <a href="#">Exportar uma API REST do API Gateway</a> . | 18 de dezembro de 2015 |
| Mapeando o corpo da solicitação ou da resposta ou os campos JSON do corpo para parâmetros de solicitação ou resposta. | Mapeie o corpo da solicitação de método ou seus campos JSON para o caminho, a string de consulta ou os cabeçalhos da solicitação de integração. Mapeie o corpo da resposta de integração ou seus campos JSON para cabeçalhos da resposta de solicitação. Para obter mais informações, consulte <a href="#">Exemplos de mapeamento de parâmetros para APIs REST no API Gateway</a> .                                                                                                                   | 18 de dezembro de 2015 |
| Trabalhar com variáveis de estágio no Amazon API Gateway                                                              | Saiba como associar atributos de configuração a um estágio de implantação de uma API no Amazon API Gateway. Para obter mais informações, consulte <a href="#">Usar variáveis de estágio para uma API REST no API Gateway</a> .                                                                                                                                                                                                                                                                        | 5 de novembro de 2015  |
| Como: habilitar o CORS para um método                                                                                 | Agora, é mais fácil habilitar o CORS (compartilhamento de recursos entre origens) para métodos no Amazon API Gateway. Para obter mais informações, consulte <a href="#">CORS para APIs REST no API Gateway</a> .                                                                                                                                                                                                                                                                                      | 3 de novembro de 2015  |
| Como: usar a autenticação SSL de cliente                                                                              | Use o Amazon API Gateway para gerar certificados SSL que podem ser usados para autenticar chamadas para o seu backend HTTP. Para obter mais informações, consulte <a href="#">Gerar e configurar um certificado SSL para autenticação de backend no API Gateway</a> .                                                                                                                                                                                                                                 | 22 de setembro de 2015 |

| Alteração                          | Descrição                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Alterado em           |
|------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|
| Integração simulada de métodos     | Saiba como <a href="#">simular a integração de uma API ao Amazon API Gateway</a> . Esse recurso permite que os desenvolvedores gerem respostas de API no API Gateway diretamente, sem necessidade de um backend de integração final antecipado.                                                                                                                                                                                                                       | 1 de setembro de 2015 |
| Suporte ao Amazon Cognito Identity | O Amazon API Gateway expandiu o escopo da variável <code>\$context</code> para que ele agora retorne informações sobre o Amazon Cognito Identity quando as solicitações são assinadas com credenciais do Amazon Cognito. Além disso, adicionamos uma variável <code>\$util</code> para escapar caracteres em JavaScript e codificar URLs e strings. Para obter mais informações, consulte <a href="#">Variáveis para transformações de dados para o API Gateway</a> . | 28 de agosto de 2015  |
| Integração com o Swagger           | Use a <a href="#">ferramenta de importação Swagger no GitHub</a> para importar as definições de API do Swagger para o Amazon API Gateway. Saiba mais sobre <a href="#">Extensões OpenAPI para o API Gateway</a> para criar e implantar APIs e métodos usando a ferramenta de importação. Com a ferramenta de importação do Swagger, você também pode atualizar APIs existentes.                                                                                       | 21 de julho de 2015   |
| Referência a modelos de mapeamento | Leia sobre o parâmetro <code>\$input</code> e as suas funções em <a href="#">Variáveis para transformações de dados para o API Gateway</a> .                                                                                                                                                                                                                                                                                                                          | 18 de julho de 2015   |
| Lançamento público inicial         | Esta é a versão pública inicial do Guia do desenvolvedor do API Gateway.                                                                                                                                                                                                                                                                                                                                                                                              | 9 de julho de 2015    |