
Amazon API Gateway

Guia do desenvolvedor



Amazon API Gateway: Guia do desenvolvedor

Copyright © 2019 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

O que é o Amazon API Gateway?	1
Arquitetura do API Gateway	1
Recursos do API Gateway	2
Casos de uso do API Gateway	3
Usar o API Gateway para criar APIs REST	3
Use o API Gateway para criar APIs WebSocket	3
Quem usa o API Gateway?	4
Acessar o API Gateway	5
Parte da infraestrutura sem servidor da AWS	5
Como começar a usar o Amazon API Gateway	5
Conceitos do API Gateway	5
Definição de preço do API Gateway	9
Conformidade do API Gateway	9
PCI DSS	9
HIPAA	10
Conceitos básicos do API Gateway	11
Pré-requisitos: prepare-se para compilar uma API no API Gateway	11
Cadastrar-se em uma conta da AWS	11
Criar um usuário administrador do IAM	11
Criar uma API REST com integrações do Lambda	12
Etapa 1: criar uma função Lambda no console do Lambda	12
Etapa 2: criar uma API REST no console do API Gateway	12
Etapa 3: implantar a API REST no console do API Gateway	15
Etapa 4: criar uma segunda função Lambda no console do Lambda	15
Etapa 5: adicionar um recurso, um método e um parâmetro à API REST no console do API Gateway	16
Próximas etapas	18
Criar uma API REST com uma integração simulada	19
Etapa 1: criar a API	19
Etapa 2: criar a integração simulada	22
Etapa 3: definir a resposta bem-sucedida	23
Etapa 4: adicionar um código de status HTTP 500 e uma mensagem de erro	23
Etapa 5: testar a integração simulada	24
Próximas etapas	25
Vídeos do Amazon API Gateway	26
Vídeos do Amazon API Gateway da série de eventos Compilação na AWS do Twitch	26
Outros vídeos do Amazon API Gateway	26
Tutoriais	28
Criar uma API com integração à Lambda	28
TUTORIAL: API Hello World com integração de proxy do Lambda	29
TUTORIAL: Criar uma API com integração de proxy do Lambda entre contas	34
TUTORIAL: Criar uma API do Lambda com integração não proxy	36
TUTORIAL: Crie uma API REST importando um exemplo	45
Criar uma API com integração HTTP	54
TUTORIAL: Criar uma API com integração de proxy HTTP	54
TUTORIAL: Criar uma API com integração não proxy HTTP	59
TUTORIAL: Criar uma API com integração privada	89
TUTORIAL: Criar uma API com integração da AWS	91
Pré-requisitos	92
Etapa 1: criar o recurso	92
Etapa 2: criar o método GET	93
Etapa 3: criar a função de execução do proxy de serviço da AWS	93
Etapa 4: especificar configurações de método e testar o método	94
Etapa 5: implantar a API	95

Etapa 6: testar a API	95
Etapa 7: Limpeza	96
TUTORIAL: API docalc com três integrações	96
Crie uma conta do AWS	97
Criar uma função do IAM assumível	97
Crie uma função do Lambda Calc	99
Testar a função do Lambda Calc	99
Criar uma API Calc	101
Integração 1: Criar um método <code>GET</code> com parâmetros de consulta para chamar a função do Lambda	101
Integração 2: Criar um método <code>POST</code> com uma carga JSON para chamar a função do Lambda	104
Integração 3: Criar um método <code>GET</code> com parâmetros de caminho para chamar a função do Lambda	106
Definições do OpenAPI de uma API de amostra para função Lambda	112
TUTORIAL: Criar uma API REST como um proxy do Amazon S3 no API Gateway	116
Configurar permissões do IAM para a API invocar ações do Amazon S3	117
Criar recursos de API para representar recursos do Amazon S3	118
Exponha um método de API para listar os buckets do Amazon S3 do chamador	119
Expor métodos de API para acessar um bucket do Amazon S3	125
Expor métodos de API para acessar um objeto do Amazon S3 em um bucket	128
Chamar a API usando um cliente de API REST	131
Definições do OpenAPI de uma API de exemplo como um proxy do Amazon S3	133
TUTORIAL: Criar uma API REST como um proxy do Amazon Kinesis	142
Criar uma função e política IAM para a API para acessar o Kinesis	143
Começar a criar uma API como um proxy do Kinesis	145
Listar fluxos do Kinesis	145
Criar, descrever e excluir um fluxo no Kinesis	150
Obter registros de e adicionar registros a um fluxo no Kinesis	156
Definições do OpenAPI de uma API de exemplo como um proxy do Kinesis	167
Criar, implantar e invocar uma API REST	182
Criação de uma API REST	182
Escolher um tipo de endpoint da API	183
Inicializar a configuração da API REST	184
Configurar métodos de API REST	204
Configurar integrações da API REST	218
Configurar respostas do gateway	263
Configurar mapeamentos de dados	270
Oferecer suporte a cargas úteis binárias	316
Habilitar compactação de carga	339
Ativar a validação de solicitação	343
Importar uma API REST	356
Controlar e gerenciar acesso a uma API REST	361
Usar políticas de recursos do API Gateway	362
Usar permissões do IAM	378
Usar políticas de VPC endpoint para APIs privadas	394
Usar tags para controlar o acesso a uma API REST	396
Usar autorizadores do Lambda	396
Use o Grupo de usuários do Cognito como um autorizador para uma API REST	414
Habilitar o CORS para um recurso da API REST	423
Gerar e configurar um certificado SSL para autenticação de back-end	429
Use o AWS WAF para proteger sua API contra explorações comuns da web	449
Utilização de planos de uso com chaves de API	450
Documentação de uma API REST	466
Representação da documentação da API no API Gateway	466
Documentar uma API usando o console do API Gateway	474
Publicar a documentação da API usando o console do API Gateway	483
Documentar uma API usando a API REST do API Gateway	483

Publicar a documentação da API usando a API REST do API Gateway	498
Importar a documentação da API	505
Controlar o acesso à documentação da API	509
Atualizar e manter uma API REST	510
Alterar um tipo de endpoint de API pública ou privada	511
Manter uma API usando o console	513
Associar ou desassociar um VPC Endpoint de uma API REST privada	515
Implantando uma API REST	516
Implantar uma API REST	518
Configurar um estágio	520
Gerar um SDK para uma API REST no API Gateway	548
Invocação de uma API REST	563
Obter a URL de invocação de uma API no console do API Gateway	563
Usar o console para testar um método de API REST	564
Usar o Postman para chamar uma API REST	565
Chamar API REST por meio dos SDKs gerados	565
Chamar a API REST por meio da biblioteca JavaScript AWS Amplify	584
Como invocar uma API privada	584
Rastreamento, registro em logs e monitoramento de uma API	586
Rastrear execução da API com o AWS X-Ray	587
Registrar chamadas para APIs do Amazon API Gateway com AWS CloudTrail	597
Monitorar a execução da API com o Amazon CloudWatch	599
Usar o Kinesis Data Firehose com registro de acesso em logs do API Gateway	605
Extensões do OpenAPI para APIs REST	606
x-amazon-apigateway-any-method	607
x-amazon-apigateway-api-key-source	608
x-amazon-apigateway-auth	608
x-amazon-apigateway-authorizer	609
x-amazon-apigateway-authtype	612
x-amazon-apigateway-binary-media-type	613
x-amazon-apigateway-documentation	613
x-amazon-apigateway-endpoint-configuration	614
x-amazon-apigateway-gateway-responses	615
x-amazon-apigateway-gateway-responses.gatewayResponse	615
x-amazon-apigateway-gateway-responses.responseParameters	616
x-amazon-apigateway-gateway-responses.responseTemplates	617
x-amazon-apigateway-integration	617
x-amazon-apigateway-integration.requestTemplates	621
x-amazon-apigateway-integration.requestParameters	622
x-amazon-apigateway-integration.responses	623
x-amazon-apigateway-integration.response	624
x-amazon-apigateway-integration.responseTemplates	625
x-amazon-apigateway-integration.responseParameters	626
x-amazon-apigateway-request-validator	626
x-amazon-apigateway-requestValidators	627
x-amazon-apigateway-requestValidators.requestValidator	628
Criar, implantar e invocar uma API WebSocket	629
Sobre APIs WebSocket	629
Gerenciamento de usuários conectados e aplicativos do cliente	630
Como invocar a integração de seu back-end	631
Envio de dados dos serviços de back-end para clientes conectados	634
Criar uma API WebSocket	634
Criar uma API WebSocket usando comandos da AWS CLI	634
Criar uma API WebSocket usando o console do API Gateway	635
Configurar rotas da API WebSocket	635
Configurar rotas	635
Especifique as configurações de solicitação de rota	636

Configurar integrações da API WebSocket	637
Configurar uma solicitação de integração da API WebSocket	637
Configurar respostas de integração da API WebSocket	640
Configurar respostas de rota da API WebSocket	642
Configurar uma resposta de rota usando o console do API Gateway	642
Configurar uma resposta de rota usando o console da AWS CLI	643
Implantar uma API WebSocket	643
Criar uma implantação de API WebSocket usando a AWS CLI	644
Criar uma implantação de API WebSocket usando o console do API Gateway	645
Invoke uma API WebSocket	645
Use wscat para se conectar a uma API WebSocket e enviar mensagens a ela	645
Use os comandos @connections em seu serviço de back-end	646
Controlar o acesso à API WebSocket	647
Utilizar autorização do IAM	647
Criar uma função do autorizador REQUEST do Lambda	648
Monitorar a execução da API WebSocket com CloudWatch	650
Expressões de seleção do WebSocket	652
.....	652
.....	654
.....	654
.....	654
.....	654
.....	654
.....	654
.....	655
Referência de modelos de mapeamento WebSocket	659
Publicar suas APIs	664
Usar o Portal do desenvolvedor sem servidor para catalogar suas APIs	664
Criar um portal do desenvolvedor	665
Configurações do portal do desenvolvedor	666
Criar um usuário administrador para seu portal de desenvolvedor	668
Publicar uma API gerenciada pelo API Gateway no portal do desenvolvedor	668
Atualizar ou excluir uma API gerenciada pelo API Gateway	669
Para remover uma API não gerenciada pelo API Gateway	670
Publicar uma API não gerenciada pelo API Gateway no portal do desenvolvedor	670
Como seus clientes podem usar o portal do desenvolvedor	671
Melhores práticas para portais do desenvolvedor	673
Vender suas APIs como SaaS	673
Inicializar a integração do AWS Marketplace com o API Gateway	674
Gerenciar a assinatura do cliente para planos de uso	675
Configurar um nome de domínio personalizado de API	676
Nomes de domínio personalizados curinga	679
Preparar certificados para uso no AWS Certificate Manager	679
Escolher uma versão mínima do TLS para um domínio personalizado	682
Como criar um nome de domínio personalizado otimizado para fronteiras	685
Configurar um nome de domínio personalizado regional para a API WebSocket ou REST	692
Migrar nomes de domínios personalizados	697
Exportar uma API REST	700
Solicitar a exportação de uma API REST	701
Fazer download da definição da API REST do OpenAPI em JSON	701
Fazer download da definição da API REST do OpenAPI em YAML	702
Fazer download da definição da API REST do OpenAPI com extensões Postman em JSON	702
Fazer download da definição da API REST do OpenAPI com integração ao API Gateway em YAML	703
Exportar a API REST usando o console do API Gateway	703
Definir uma implantação da versão Canary	704
Implantação da versão Canary no API Gateway	704

Criar uma implantação da versão Canary	705
Atualizar uma versão canary	709
Promover uma versão canary	711
Desativar uma versão canary	713
Monitorar a configuração de sua API	715
Tipos de recursos com suporte	715
Configuração de AWS Config	716
Configuração do AWS Config para registrar recursos do API Gateway	716
Visualizar detalhes de configuração do API Gateway no console do AWS Config	716
Avaliar recursos do API Gateway usando regras do AWS Config	717
Marcar seus recursos do API Gateway	718
Recursos do API Gateway que podem ser marcados	718
Herança de tags na API V1 do Amazon API Gateway	719
Restrições de tags e convenções de uso	720
Controle de acesso baseado em tags	720
Exemplo 1: limitar ações com base em tags de recursos	721
Exemplo 2: limitar ações com base em tags na solicitação	721
Exemplo 3: negar ações com base em tags de recurso	722
Exemplo 4: permitir ações com base em tags de recursos	723
Referências de API na V1 e V2 do API Gateway	724
Limites e observações importantes	725
Limites do API Gateway	725
Limites do nível da conta do API Gateway	725
Limites do API Gateway para configurar e executar uma API WebSocket	726
Limites do API Gateway para configurar e executar uma API REST	726
Limites do API Gateway para criação, implantação e gerenciamento de uma API	728
Observações importantes	730
Observações importantes para APIs REST e WebSocket	730
Observações importantes para APIs WebSocket	731
Observações importantes para APIs REST	731
Histórico do documento	735
Atualizações anteriores	739
AWS Glossary	746

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 WebSocket em qualquer escala. Os desenvolvedores de API podem criar APIs que acessem a AWS ou outros serviços da web, bem como dados armazenados na [Nuvem AWS](#). Como um desenvolvedor de API do API Gateway, você pode criar APIs para uso em seus próprios aplicativos clientes (apps). 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?” \(p. 4\)](#).

O API Gateway cria APIs REST que:

- São baseadas em HTTP.
- Seguem o protocolo [REST](#), que permite a comunicação entre cliente e servidor sem estado.
- Implementem os métodos HTTP padrão, como GET, POST, PUT, PATCH e DELETE.

Para obter mais informações sobre APIs REST do API Gateway, consulte [the section called “Usar o API Gateway para criar APIs REST” \(p. 3\)](#) e [the section called “Criação de uma API REST” \(p. 182\)](#).

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.

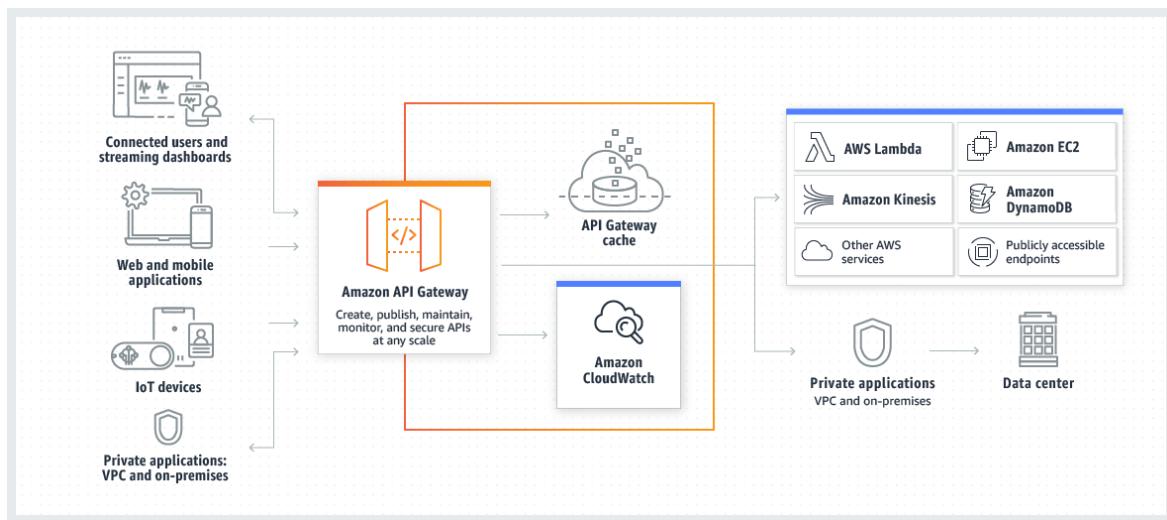
Para obter mais informações sobre APIs WebSocket do API Gateway, consulte [the section called “Use o API Gateway para criar APIs WebSocket” \(p. 3\)](#) e [the section called “Sobre APIs WebSocket” \(p. 629\)](#).

Tópicos

- [Arquitetura do API Gateway \(p. 1\)](#)
- [Recursos do API Gateway \(p. 2\)](#)
- [Casos de uso do API Gateway \(p. 3\)](#)
- [Acessar o API Gateway \(p. 5\)](#)
- [Parte da infraestrutura sem servidor da AWS \(p. 5\)](#)
- [Como começar a usar o Amazon API Gateway \(p. 5\)](#)
- [Conceitos do Amazon API Gateway \(p. 5\)](#)
- [Definição de preço do API Gateway \(p. 9\)](#)
- [Conformidade do API Gateway \(p. 9\)](#)

Arquitetura do API Gateway

O diagrama a seguir mostra a arquitetura do API Gateway.



Esse diagrama ilustra como as APIs compiladas no Amazon API Gateway fornecem a você ou aos seus clientes desenvolvedores uma experiência de desenvolvedor integrada e consistente para a compilação de aplicativos sem servidor da AWS. O API Gateway processa todas as tarefas envolvidas na aceitação e no processamento de centenas de milhares de chamadas de API simultâneas, incluindo gerenciamento de tráfego, autorização e controle de acesso, monitoramento e gerenciamento de versão da API. O API Gateway atua como uma "porta de entrada" para os aplicativos acessarem dados, lógica de negócios ou funcionalidade de seus serviços de back-end, como cargas de trabalho em execução no Amazon Elastic Compute Cloud (Amazon EC2), código em execução no AWS Lambda, qualquer aplicativo web ou aplicativos de comunicação em tempo real.

Recursos do API Gateway

O Amazon API Gateway oferece recursos como os seguintes:

- Suporte para APIs ([WebSocket \(p. 629\)](#)) com estado e ([REST \(p. 182\)](#)) sem estado.
- Mecanismos de [autenticação \(p. 361\)](#) poderosos e flexíveis, como políticas do AWS Identity and Access Management, funções do autorizador do Lambda e grupos de usuários do Amazon Cognito.
- [Portal do desenvolvedor \(p. 664\)](#) para publicar suas APIs.
- [Implantações de versão Canary \(p. 704\)](#) para lançar alterações com segurança.
- Registro em log e monitoramento do [CloudTrail \(p. 597\)](#) 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 “Monitorar a execução da API com o Amazon CloudWatch” \(p. 599\)](#) e [the section called “Monitorar a execução da API WebSocket com CloudWatch” \(p. 650\)](#).
- Capacidade de usar modelos do AWS 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 \(p. 676\)](#).
- Integração ao [AWS WAF \(p. 449\)](#) para proteger suas APIs contra explorações comuns da web.
- Integração com latências de desempenho [AWS X-Ray \(p. 587\)](#) para compreensão e triagem.

Para obter uma lista completa de lançamentos de recursos do API Gateway, consulte [Histórico do documento \(p. 735\)](#).

Casos de uso do API Gateway

Tópicos

- [Usar o API Gateway para criar APIs REST \(p. 3\)](#)
- [Use o API Gateway para criar APIs WebSocket \(p. 3\)](#)
- [Quem usa o API Gateway? \(p. 4\)](#)

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 chamador, e um método `GET /expenses` pode consultar as despesas relatadas incorridas pelo chamador.

O aplicativo não precisa saber onde os dados solicitados são armazenados e de onde são obtidos no back-end. 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 back-end usando as solicitações de integração e respostas de integração.

Por exemplo, com o DynamoDB como o back-end, o desenvolvedor da API configura a solicitação de integração para encaminhar a solicitação do método de entrada para o back-end 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 back-end 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, converta o formato dos dados de saída do back-end 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, como:

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

Use o API Gateway para criar APIs WebSocket

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

Por exemplo, você pode criar um aplicativo sem servidor usando uma API WebSocket do API Gateway e AWS Lambda para enviar e receber mensagens de e para usuários individuais ou grupos de usuários em uma sala de bate-papo. Ou você pode invocar serviços de back-end, como AWS Lambda, Amazon Kinesis ou um endpoint HTTP com base no conteúdo da mensagem.

Você pode usar as APIs WebSocket do API Gateway para criar aplicativos 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 da 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 back-end
- 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 aplicativos.

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 do IAM na conta da AWS que é a proprietária da API.

Um desenvolvedor de aplicativos cria um aplicativo 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 aplicativos não precisa ter uma conta da AWS, contanto que a API não exija permissões do IAM ou 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 da API, nomeado `apigateway`, para criar, configurar e implantar uma API. Cada API inclui um conjunto de recursos e métodos. Um recurso é uma entidade lógica que um aplicativo pode acessar através de um caminho de recurso.

Como um desenvolvedor de APIs, você pode criar e gerenciar uma API usando o console do API Gateway, descrito em [Conceitos básicos sobre o Amazon API Gateway \(p. 11\)](#) ou chamando [Referências de API na V1 e V2 do API Gateway \(p. 724\)](#). Há várias maneiras de chamar essa API. Elas incluem o uso da Interface da linha de comando (ILC) da AWS ou o uso de um SDK da AWS. Além disso, você pode habilitar a criação da API com [modelos do AWS CloudFormation](#) ou [Extensões do API Gateway para OpenAPI \(p. 606\)](#) (no caso de APIs REST). Para obter uma lista das regiões em que o API Gateway está disponível, bem como os endpoints de serviço de controle associados, consulte [Regiões e endpoints](#).

Como chamar uma API do API Gateway

Um desenvolvedor de aplicativos trabalha com o componente de serviço do API Gateway para execução da API, chamado `execute-api`, para invocar uma API que foi 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. Você pode usar o console do API Gateway para testar a invocação da API. Para APIs REST, você pode usar um cliente de API REST, como [cURL](#) ou [POSTMAN](#) ou um SDK gerado pelo API Gateway para a API invocar a API.

Acessar o API Gateway

Você pode acessar o Amazon API Gateway das seguintes maneiras:

- Console de gerenciamento da AWS – Os procedimentos em todo este guia explicam como usar o Console de gerenciamento da AWS para realizar tarefas.
- SDKs da AWS – Se 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 obter mais informações, consulte [Ferramentas da Amazon Web Services](#).
- APIs V1 e V2 do API Gateway – Se você está usando uma linguagem de programação para a qual não há um 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 o tópico sobre [Noções básicas de configuração com o AWS Command Line Interface](#), no Guia do usuário do AWS Command Line Interface.
- AWS Tools para Windows PowerShell–Para obter mais informações, consulte o tópico sobre como [Configurar o AWS Tools para Windows PowerShell](#) no Guia do usuário do AWS Tools para Windows PowerShell.

Parte da infraestrutura sem servidor da AWS

Juntamente com o [AWS Lambda](#), o API Gateway forma a parte voltada para o aplicativo da infraestrutura sem servidor da AWS. Para que um aplicativo 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 aplicativos sem servidor, o API Gateway oferece suporte a [integrações de proxy simplificadas \(p. 222\)](#) com endpoints do AWS Lambda e HTTP.

Como começar a usar o Amazon API Gateway

Para obter uma rápida introdução ao Amazon API Gateway, consulte os seguintes tópicos:

- [Conceitos básicos do API Gateway \(p. 11\)](#), que fornece demonstrações e vídeos que informam como criar APIs usando o Amazon API Gateway.
- [Anúncio de APIs WebSocket no Amazon API Gateway](#), que mostra como criar uma API WebSocket.

Conceitos do Amazon API Gateway

API Gateway

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

- Crie, implante e gerencie uma interface de programação de aplicativo (API) [REST](#) para expor endpoints HTTP de back-end, funções do AWS Lambda ou outros serviços da AWS.
- Crie, implante e gerencie 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 recursos e métodos HTTP que são integrados com endpoints HTTP de back-end, funções do Lambda ou outros serviços da AWS. Essa coleção pode ser implantada 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.

API WebSocket do API Gateway

Uma coleção de rotas do WebSocket e de chaves de rota que são integradas com endpoints HTTP de back-end, funções do Lambda ou outros serviços da AWS. A coleção pode ser implantada 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

Snapshot de um ponto no tempo 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

Conta da AWS que tem uma implantação do API Gateway (por exemplo, um provedor de serviços que também é compatível com acesso programático).

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 \(p. 7\)](#)
- [Endpoint privado de API \(p. 8\)](#)
- [Endpoint de API regional \(p. 8\)](#)

Chave da API

Uma string alfanumérica que o API Gateway usa para identificar um desenvolvedor de aplicativos que usa sua API REST ou WebSocket. O API Gateway pode gerar chaves da API em seu nome ou você pode importá-las de um arquivo CSV. Você pode usar chaves da API com [autorizadores do Lambda \(p. 396\)](#) ou [planos de uso \(p. 450\)](#) para controlar o acesso às APIs.

Consulte [Endpoints de API \(p. 6\)](#).

Proprietário de API

Consulte [Desenvolvedor de APIs \(p. 6\)](#).

Estágio de API

Uma referência lógica a um estado do ciclo de vida de sua API REST ou WebSocket (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 \(p. 6\)](#).

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 back-end.

Portal do desenvolvedor

Um aplicativo que permite que os clientes se inscrevam, descubram e assinem seus produtos de API (planos de uso de produtos do API Gateway), gerenciem suas chaves de API e visualizem suas métricas de uso para as APIs.

Endpoint de API otimizada para fronteiras

O nome do host padrão de uma API do API Gateway implantada para a região especificada enquanto usa uma distribuição do CloudFront para facilitar o acesso geral de clientes nas 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 \(p. 6\)](#).

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 back-end.

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 back-end para o formato de resposta que é retornado a um aplicativo cliente.

modelo de mapeamento

Scripts em [Velocity Template Language \(VTL\)](#) que transformam um corpo de solicitação do formato de dados do front-end para o formato de dados do back-end, ou que transformam um corpo de resposta do formato de dados do back-end para o formato de dados do 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 back-end para uma solicitação. O mesmo aplica-se para uma resposta, na qual a carga é passada do back-end ao cliente.

Solicitação de método

A interface pública de um método de API REST no API Gateway que define os parâmetros e o corpo que um desenvolvedor de aplicativos deve enviar nas solicitações para acessar o back-end 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 back-end de integração. Como desenvolvedor de API, 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.

API privado

Consulte [Endpoint privado de API \(p. 8\)](#).

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 de uma VPC do cliente por meio de um endpoint de API REST privada sem expor os recursos à Internet pública.

Integração de proxy

Uma configuração de integração do API Gateway simplificada para uma API REST ou WebSocket. Você pode configurar uma integração de proxy como um tipo de integração de proxy HTTP ou uma integração de proxy Lambda. Para a integração de proxy HTTP, o API Gateway passa toda a solicitação e a resposta entre o front-end e um back-end HTTP. Para a integração de proxy do Lambda, o API Gateway envia a solicitação inteira como entrada para uma função do Lambda de back-end. Em seguida, 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, que é representado por uma variável de caminho voraz (por exemplo, `{proxy+}`), combinada com um método genérico ANY.

Endpoint de API regional

O nome do host de uma API implantada na região especificada e que pretende atender clientes, como instâncias do EC2, na mesma região da AWS. As solicitações da API são direcionadas diretamente para o API Gateway específico da região sem passar por qualquer distribuição do CloudFront. Para solicitações em uma mesma região, um endpoint regional ignora a ida e volta desnecessária para uma distribuição do CloudFront. Além disso, você pode aplicar o [roteamento baseado em latência](#) a 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 com latência mais baixa.

Consulte [Endpoints de API \(p. 6\)](#).

Rota

Uma rota do 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 back-end 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 back-end 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 back-end 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 back-end 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 \(p. 450\)](#) 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 clientes e o API Gateway. Não há conexão persistente entre o API Gateway e as integrações de back-end, como funções do Lambda. Os serviços de back-end são invocados, conforme necessário, com base no conteúdo de mensagens recebidas de clientes.

Definição de preço do API Gateway

Para obter informações gerais sobre preços específicos à região do API Gateway, consulte [Definição de preço do Amazon API Gateway](#).

A tabela a seguir lista as exceções ao esquema geral de definição de preço:

- O armazenamento em cache de APIs no Amazon API Gateway não está qualificado para o nível gratuito da AWS.
- Métodos de chamada com [tipo de autorização](#) de AWS_IAM, CUSTOM e COGNITO_USER_POOLS não são cobrados por falhas de autorização e autenticação.
- Chamadas para métodos que exigem chaves de API não são cobrados quando as chaves de API estão ausentes ou são inválidas.
- As solicitações limitadas pelo API Gateway não serão cobradas quando a taxa de solicitação ou de intermitência exceder os limites pré-configurados.
- As solicitações limitadas pelo plano de uso não serão cobradas quando os limites de taxa ou a cota exceder os limites pré-configurados.

Conformidade do API Gateway

Para obter informações sobre a conformidade do API Gateway com vários regulamentos de conformidade de segurança e padrões de auditoria, consulte as páginas a seguir:

- [Conformidade da Nuvem AWS](#)
- [Serviços da AWS no escopo pelo programa de conformidade](#)

Além disso, consulte as seções a seguir para obter detalhes sobre como o API Gateway está em conformidade com os padrões PCI DSS e HIPAA.

PCI DSS

API Gateway oferece suporte a processamento, armazenamento e transmissão de dados de cartão de crédito por um comerciante ou provedor de serviços, e foi validado como em conformidade com o Data Security Standard (DSS – Padrão de segurança de dados) da Payment Card Industry (PCI – Setor de cartão de crédito). Para obter mais informações sobre PCI DSS, incluindo como solicitar uma cópia do pacote de conformidade com PCI da AWS, consulte [Nível 1 do PCI DSS](#).

HIPAA

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](#).

Conceitos básicos sobre o Amazon API Gateway

Para ajudar você a começar a usar o Amazon API Gateway, analise as seguintes demonstrações práticas e os outros recursos:

- [Pré-requisitos: prepare-se para compilar uma API no API Gateway](#) aborda como criar uma conta da AWS e configurar um usuário administrador do IAM.
- [Criar uma API REST com integrações do Lambda](#) mostra como criar APIs REST com funções do Lambda no back-end. As duas técnicas de integração de proxy e não proxy estão inclusas.
- [Criar uma API REST com uma integração simulada](#) mostra como criar uma API REST sem um back-end real.
- [TUTORIAL: criar uma API REST importando um exemplo](#) mostra como criar uma API de exemplo importando um arquivo OpenAPI 2.0.
- [Criar uma API WebSocket](#) mostra como criar uma API WebSocket no API Gateway.
- [Vídeos do Amazon API Gateway](#) fornece links para vídeos que abordam os conceitos básicos do API Gateway.

Pré-requisitos: prepare-se para compilar uma API no API Gateway

Tópicos

- [Cadastrar-se em uma conta da AWS \(p. 11\)](#)
- [Criar um usuário administrador do IAM \(p. 11\)](#)

Antes de usar o Amazon API Gateway pela primeira vez, você deve ter uma conta da AWS.

Em muitos dos tutoriais em [Conceitos básicos do API Gateway \(p. 11\)](#) e [Tutoriais \(p. 28\)](#), as políticas necessárias do IAM são criadas para você. No entanto, quando começar a criar sua própria API, você precisará estudar as informações abaixo e em [the section called “Usar permissões do IAM” \(p. 378\)](#).

Cadastrar-se em uma conta da AWS

Para usar o Amazon API Gateway, o AWS Lambda e outros serviços da AWS, você precisa de uma conta da AWS. Se você não tiver uma conta, acesse aws.amazon.com e escolha Create an AWS Account. Para obter instruções detalhadas, consulte [Criar e ativar uma conta da AWS](#).

Criar um usuário administrador do IAM

Como prática recomendada, você também deve criar um usuário do AWS Identity and Access Management (IAM) com permissões de administrador. Use isso para todo o trabalho que não requer credenciais raiz. Crie uma senha para acesso ao console e chaves de acesso para usar ferramentas da linha de comando. Para obter instruções, consulte [Criar seu primeiro usuário administrador e o grupo do IAM](#) no Guia do usuário do IAM.

Criar uma API REST com integrações do Lambda no Amazon API Gateway

Você pode usar essa demonstração para criar e implantar uma API REST com uma integração de proxy do Lambda e uma integração não proxy do Lambda no Amazon API Gateway.

Em uma integração do Lambda, a solicitação do método HTTP do cliente é mapeada para uma [chamada de função do Lambda de back-end](#).

Em uma integração de proxy do Lambda, toda a solicitação do cliente é enviada para a função do Lambda de back-end como está, exceto pelo fato de que a ordem dos parâmetros da solicitação não é preservada. O API Gateway mapeia toda a solicitação do cliente para o parâmetro `event` de entrada da função do Lambda de back-end. A saída da função do Lambda, inclusive o código de status, os cabeçalhos e o corpo, é retornada ao cliente como está. Para muitos casos de uso, esse é o tipo de integração preferencial. Para obter mais informações, consulte [the section called “Configurar integrações de proxy do Lambda” \(p. 227\)](#).

Em uma integração não proxy do Lambda (também chamada de "integração personalizada"), você configura a maneira como os parâmetros, os cabeçalhos e o corpo da solicitação do cliente são convertidos no formato exigido pela função do Lambda de back-end. Você configura a maneira como a saída da função do Lambda é convertida de volta para o formato exigido pelo cliente. Para obter mais informações, consulte [the section called “Configurar integrações personalizadas do Lambda” \(p. 239\)](#).

Dependendo do seu caso de uso, você pode optar por usar a integração de proxy do Lambda, a integração não proxy do Lambda ou as duas na sua API do API Gateway.

Tópicos

- [Etapa 1: criar uma função Lambda no console do Lambda \(p. 12\)](#)
- [Etapa 2: criar uma API REST no console do API Gateway \(p. 12\)](#)
- [Etapa 3: implantar a API REST no console do API Gateway \(p. 15\)](#)
- [Etapa 4: criar uma segunda função Lambda no console do Lambda \(p. 15\)](#)
- [Etapa 5: adicionar um recurso, um método e um parâmetro à API REST no console do API Gateway \(p. 16\)](#)
- [Próximas etapas \(p. 18\)](#)

Etapa 1: criar uma função Lambda no console do Lambda

Nesta etapa, você usa o console do AWS Lambda para criar uma função do Lambda simples. Você usará essa função nas etapas a seguir.

1. Se você ainda não tiver feito isso, conclua as etapas em [the section called “Pré-requisitos: prepare-se para compilar uma API no API Gateway” \(p. 11\)](#).
2. Conclua as etapas em [Para criar uma função Lambda](#) no AWS Lambda Developer Guide.

Etapa 2: criar uma API REST no console do API Gateway

Nesta etapa, você cria uma API REST simples no console do API Gateway e anexa sua função Lambda a ela como um back-end.

1. No menu Services (Serviços), escolha API Gateway para ir para o console do API Gateway.
2. Se essa for a primeira vez que você usa o API Gateway, verá uma página que apresenta os recursos do serviço. Escolha Get Started. 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).



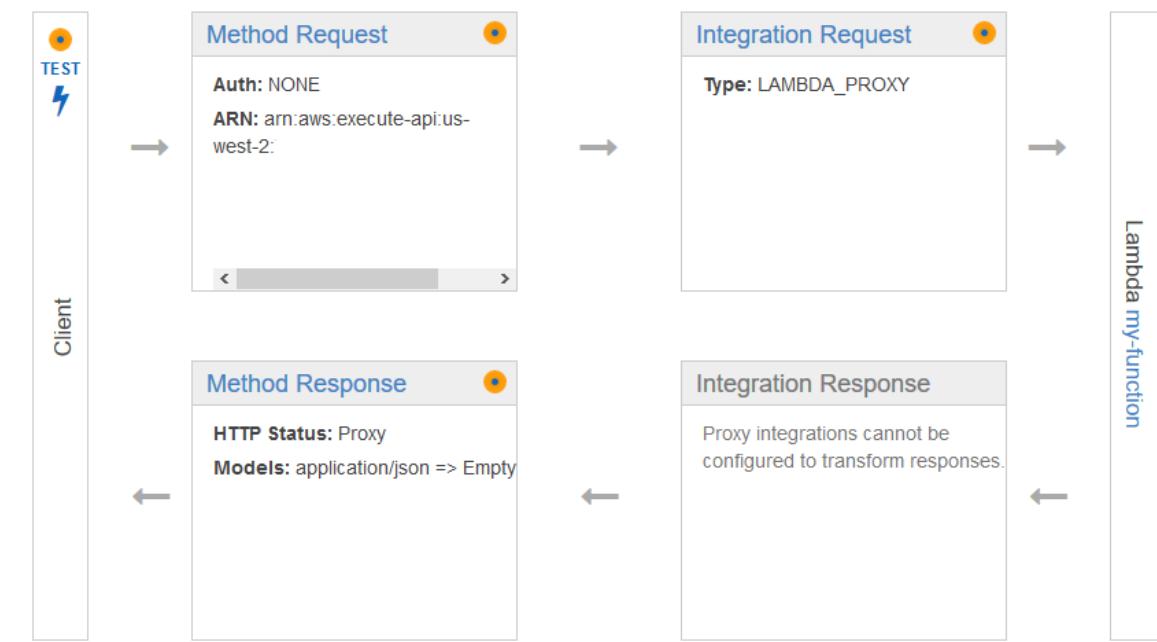
3. Em Choose the protocol (Escolher o protocolo), selecione REST.
4. Em Create new API (Criar nova API), selecione New API (Nova API).
5. Em Settings (Configurações):
 - Em API name (Nome da API), insira `my-api`.
 - Se desejar, digite uma descrição no campo Description (Descrição). Caso contrário, deixe-o vazio.
 - Deixe Endpoint Type (Tipo de endpoint) definido como Regional.
6. Escolha Create API (Criar API).
7. Em Resources (Recursos), você não verá nada além de `/`. Esse é o recurso no nível raiz, que corresponde ao URL do caminho base da sua API `https://b123abcde4.execute-api.us-west-2.amazonaws.com/{stage-name}`.

No menu suspenso Actions (Ações), escolha Create Method (Criar método).

8. Sob o nome do recurso (`/`), você verá um menu suspenso. Escolha GET e depois escolha o ícone de marca de seleção para salvar sua opção.
9. No painel `/ - GET - Setup` (`/ - GET - Configuração`), para o Integration type (Tipo de integração), escolha Lambda Function (Função Lambda).
10. Selecione Use Lambda proxy integration (Usar a integração de proxy do Lambda).
11. Em Lambda Region (Região Lambda), escolha a região na qual você criou sua função Lambda.
12. No campo Lambda Function (Função Lambda), digite qualquer caractere e escolha `my-function` (ou qualquer nome que você atribuiu à função criada na etapa anterior) no menu suspenso. (Se o menu suspenso não aparecer, exclua o caractere que você acabou de digitar para que o menu suspenso seja exibido.) Deixe Use Default Timeout (Usar o tempo limite padrão) marcado. Selecione Save (Salvar) para salvar as alterações.
13. Quando o pop-up Add Permission to Lambda Function (Adicionar permissão à função Lambda) for exibido (dizendo "You are about to give API Gateway permission to invoke your Lambda function..." (Você está prestes a conceder permissão para o API Gateway invocar sua função Lambda...)), escolha OK para conceder ao API Gateway essa permissão.

Agora, você verá um painel / – GET – Method Execution (/ – GET – Execução de método):

/ - GET - Method Execution



O painel Method Execution (Execução de método) contém estes itens, no sentido horário:

- Client (Cliente): essa caixa representa o cliente (navegador ou aplicativo) que chama o método GET da API. Se você escolher o link Test (Testar) e escolher Test (Testar), isso simulará uma solicitação GET de um cliente.
- Method Request (Solicitação de método): essa caixa representa a solicitação GET do cliente quando ela é recebida pelo API Gateway. Se você escolher Method Request (Solicitação de método), verá configurações para itens como autorização e modificação da solicitação de método antes que ela seja transmitida ao back-end como uma solicitação de integração. Para essa etapa, deixe tudo definido com os valores padrão.
- Integration Request (Solicitação de integração): essa caixa representa a solicitação GET quando ela é transmitida ao back-end. Aqui as configurações dependem do Integration Type (Tipo de integração) que está selecionado. As configurações URL Path Parameters (Parâmetros de caminho de URL), URL Query String Parameters (Parâmetros de string de consulta de URL), HTTP Headers (Cabeçalhos de HTTP) e Mapping Templates (Modelos de mapeamento) servem para modificar a solicitação de método conforme necessário pelo back-end específico. Deixe o Integration Type (Tipo de integração) definido como Lambda function (Função Lambda) e as outras configurações definidas com os valores padrão.
- Lambda **my-function**: essa caixa representa a função Lambda de back-end que você criou na Etapa 1. Se você escolher **my-function**, isso abrirá a função Lambda **my-function** no console do Lambda.
- Integration Response (Resposta da integração): essa caixa representa a resposta do back-end, antes de ser transmitida ao cliente como uma resposta de método. Para uma integração de proxy do Lambda, essa caixa inteira fica esmaecida, porque uma integração de proxy retorna a [saída da função do Lambda](#) como ela está. Para uma integração não proxy do Lambda, você pode configurar a resposta da integração para transformar a saída da função do Lambda em um formato adequado para o aplicativo cliente. Você aprenderá como fazer isso mais adiante nessa demonstração.

Para este procedimento de Conceitos básicos, deixe tudo configurado com os valores padrão.

- Method Response (Resposta do método): essa caixa representa a resposta do método retornada ao cliente como um código de status HTTP, um cabeçalho de resposta opcional e um corpo de resposta

opcional. Por padrão, o corpo da resposta que é retornado pela sua função Lambda é transmitida como está como um documento JSON, portanto, a configuração padrão do corpo da resposta é `application/json` com um modelo vazio (indicando que o corpo é transmitido como está). Aqui também, deixe tudo definido com os valores padrão.

Etapa 3: implantar a API REST no console do API Gateway

Depois de concluir a Etapa 2, você criou uma API, mas ainda não é possível usá-la. Isso ocorre porque ela precisa ser implantada.

1. No menu suspenso Ações, escolha Implantar API.
2. No menu suspenso Estágio de implantação, escolha [Novo estágio].
3. Em Stage name (Nome do estágio), insira `prod`.
4. Escolha Deploy.
5. No Stage Editor (Editor de estágio) `prod`, observe o Invoke URL (URL de invocação) na parte superior. Ele deve estar no seguinte formato: (`https://b123abcde4.execute-api.us-west-2.amazonaws.com/prod`). Se você escolher o Invoke URL (URL de invocação), ele abrirá uma nova guia do navegador com esse URL. Se você atualizar a nova guia do navegador, verá o corpo da mensagem ("Hello from Lambda! ") retornado.

Para testes reais de API, você usaria uma ferramenta como [cURL](#) ou [POSTMAN](#) para testar a API. Por exemplo, se você tiver `cURL` instalado em seu computador, poderá testar a API da seguinte maneira:

1. Abra uma janela do terminal.
2. Copie o seguinte comando `cURL` e cole-o na janela do terminal, substituindo `b123abcde4` pelo ID de API da sua API e `us-west-2` pela região em que a API foi implantada.

Linux or Macintosh

```
curl -X GET 'https://b123abcde4.execute-api.us-west-2.amazonaws.com/prod'
```

Windows

```
curl -X GET "https://b123abcde4.execute-api.us-west-2.amazonaws.com/prod"
```

Este comando retorna a seguinte saída: "Hello from Lambda!"

Etapa 4: criar uma segunda função Lambda no console do Lambda

Nesta etapa, você criará uma segunda função Lambda de back-end. Essa função utiliza um parâmetro de entrada. Na Etapa 5, você criará um recurso filho na sua API que tem seu próprio método `GET`, que será configurado para passar um valor de parâmetro para essa nova função.

1. Escolha Lambda no menu Services (Serviços) para ir para o console do Lambda.
2. Selecione Create function (Criar função).
3. Escolha Author from scratch.

4. Em Function name (Nome da função), insira `my-function2`.
5. Em Runtime (Tempo de execução), selecione Node.js 8.10 ou Python 3.7.
6. Em Permissions (Permissões), expanda Choose or create an execution role (Escolher ou criar uma função de execução). Na lista suspensa Execution role (Função de execução), escolha Use an existing role (Usar uma função existente). Em Existing role (Função existente), escolha a função da função anterior do Lambda. O nome da função estará no formato `service-role/my-function-a1b23c4d`. Ou escolha Create a new role with basic Lambda permissions (Criar uma nova função com permissões básicas do Lambda).
7. Selecione Create function (Criar função).
8. No painel Function code (Código da função), você verá a função padrão "Hello from Lambda!" (Olá do Lambda!). Substitua toda a função com o seguinte:

Node.js 8.10

```
exports.handler = async (event) => {
  let myParam = event.myParam;
  const response = {
    statusCode: 200,
    body: JSON.stringify(myParam)
  };
  return response;
};
```

Python 3.7

```
import json
def lambda_handler(event, context):
    myParam = event['myParam']
    return {
        'statusCode': 200,
        'body': json.dumps(myParam)
    }
```

9. Escolha Salvar.

Etapa 5: adicionar um recurso, um método e um parâmetro à API REST no console do API Gateway

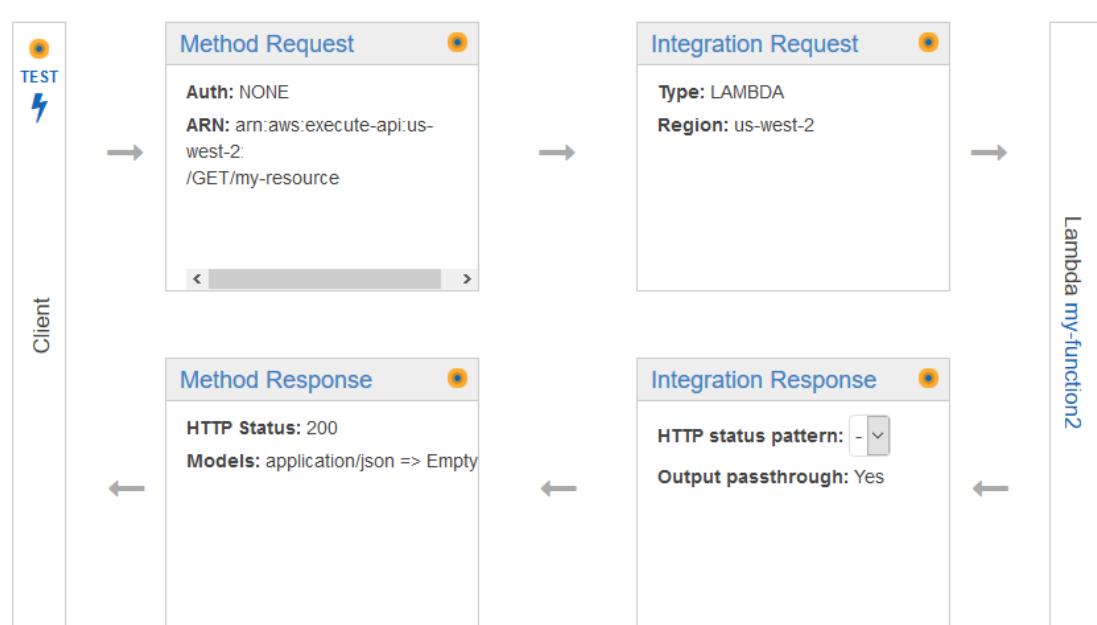
Os recursos e métodos são os substantivos e verbos da sua API REST. Nesta etapa, você criará um recurso filho para a API e adicionará um método GET ao recurso. Você adicionará um parâmetro de string de consulta ao novo método para corresponder ao parâmetro de entrada para a função Lambda criada na Etapa 4. Você integrará a nova função com o método para ilustrar como obter a entrada de usuário (uma string de texto simples "Hello from API Gateway!") e a mapeará para a entrada da função Lambda (também uma string simples).

1. Escolha API Gateway no menu Services (Serviços) para ir para o console do API Gateway.
2. Na lista APIs, escolha `my-api`.
3. Em Resources (Recursos), escolha `/`.
4. No menu suspenso Actions (Ações), escolha Create Resource (Criar recurso).
5. Em Resource Name (Nome do recurso), insira `my-resource`. Observe que o campo Resource Path (Caminho do recurso) será preenchido automaticamente com o nome do recurso.
6. Escolha Create Resource (Criar recurso).
7. No menu suspenso Actions (Ações), escolha Create Method (Criar método).

8. Sob o nome do recurso (my-resource), você verá um menu suspenso. Escolha GET e depois escolha o ícone de marca de seleção para salvar sua opção.
9. No painel /my-resource – GET – Setup (/my-resource – GET – Configuração), para o Integration type (Tipo de integração), escolha Lambda Function (Função Lambda).
10. Em Lambda Region (Região Lambda), escolha a região na qual você criou sua função Lambda.
11. Na caixa Lambda Function (Função Lambda, digite qualquer caractere e escolha my-function2 no menu suspenso. (Se o menu suspenso não aparecer, exclua o caractere que você acabou de digitar para que o menu suspenso seja exibido.) Selecione Save (Salvar) para salvar as alterações.
12. Quando o pop-up Add Permission to Lambda Function (Adicionar permissão à função Lambda) for exibido (dizendo "You are about to give API Gateway permission to invoke your Lambda function..." (Você está prestes a conceder permissão para o API Gateway invocar sua função Lambda...)), escolha OK para conceder ao API Gateway essa permissão.
13. Agora você verá um painel /my-resource – GET – Method Execution (/my-resource – GET – Execução de método). Observe que, desta vez, a caixa Integration Response (Resposta de Integração) não está esmaecida.

Escolha Integration Request (Solicitação de integração).

/my-resource - GET - Method Execution



14. Expanda Mapping Templates (Modelos de mapeamento).
15. Defina Request body passthrough (Passagem do corpo da solicitação) como When there are no templates defined (Quando não há modelos definidos).
16. Escolha Add mapping template (Adicionar modelo de mapeamento).
17. Em Content-Type (Tipo de conteúdo), insira application/json e escolha o ícone de marca de seleção para salvar sua opção.
18. Se aparecer um pop-up Change passthrough behavior (Alterar comportamento de passagem), escolha Yes, secure this integration. Isso garante que sua integração só permite solicitações que correspondem ao Content-Type (Tipo de conteúdo) recém-definido.
19. Na janela do modelo, digite o seguinte:

```
{
    "myParam": "$input.params('myParam')"
}
```

}

Esse é o documento JSON que será transmitido à função Lambda no corpo do objeto de evento de entrada. O lado direito informa ao API Gateway o que procurar na sua string de consulta de entrada. O lado esquerdo armazena no parâmetro que você está prestes a adicionar à função Lambda.

20. Escolha Save.
21. No menu suspenso Ações, escolha Implantar API.
22. No menu suspenso Estágio de implantação, escolha prod.
23. Escolha Deploy.

Teste a API da seguinte maneira:

1. Abra uma janela do terminal.
2. Copie o seguinte comando cURL e cole-o na janela do terminal, substituindo **b123abcde4** pelo ID de API da sua API e **us-west-2** pela região em que a API foi implantada.

Linux or Macintosh

```
curl -X GET 'https://b123abcde4.execute-api.us-west-2.amazonaws.com/prod'
```

Windows

```
curl -X GET "https://b123abcde4.execute-api.us-west-2.amazonaws.com/prod"
```

Isso invocará sua API e você verá "Hello from Lambda!" (Olá do Lambda!) retornado. Isso ocorre porque a nova função do Lambda, `my-function2`, não será anexada ao recurso raiz (/). Ela é anexada ao novo recurso que você criou, `my-resource`. Portanto, a função do Lambda que você acabou de invocar é o "Hello from Lambda!", a função que você criou anteriormente, `my-function`.

3. Na linha de comando cURL, após prod, digite /`my-resource?myParam=Hello%20from%20API%20Gateway!`. O comando completo deve ficar assim:

Linux or Macintosh

```
curl -X GET 'https://b123abcde4.execute-api.us-west-2.amazonaws.com/prod/my-resource?myParam=Hello%20from%20API%20Gateway!'
```

Windows

```
curl -X GET "https://b123abcde4.execute-api.us-west-2.amazonaws.com/prod/my-resource?myParam=Hello%20from%20API%20Gateway!"
```

Agora, você deve ver a resposta correta retornada: `{"statusCode": 200, "body": "\"Hello from API Gateway!\""}.`

Próximas etapas

Explore um ou todos os tópicos a seguir para continuar a se familiarizar com o Amazon API Gateway.

Para saber mais a respeito	Acessar
Transmitir a entrada para uma função do Lambda de back-end	the section called “Formato de entrada de uma função Lambda para integração de proxy” (p. 234)
Retornar a saída de uma função do Lambda de back-end	the section called “Formato de saída de uma função Lambda para integração de proxy” (p. 238)
Configurar um nome de domínio personalizado para a API	the section called “Configurar um nome de domínio personalizado de API” (p. 676)
Adicionar uma função de autorizador do Lambda à API	the section called “Usar autorizadores do Lambda” (p. 396)
Adicionar um autorizador de grupo de usuários do Amazon Cognito à API	the section called “Use o Grupo de usuários do Cognito como um autorizador para uma API REST” (p. 414)
Ativar o CORS para a API	the section called “Habilitar o CORS para um recurso da API REST” (p. 423)

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

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

Criar uma API REST com uma integração simulada no Amazon API Gateway

Você pode usar esta demonstração para criar e implantar uma API com uma integração simulada no Amazon API Gateway. Uma integração simulada permite que sua API retorne uma resposta para uma solicitação diretamente, sem enviar a solicitação para um back-end. Isso permite que você desenvolva a API independentemente do back-end.

Tópicos

- [Etapa 1: criar a API \(p. 19\)](#)
- [Etapa 2: criar a integração simulada \(p. 22\)](#)
- [Etapa 3: definir a resposta bem-sucedida \(p. 23\)](#)
- [Etapa 4: adicionar um código de status HTTP 500 e uma mensagem de erro \(p. 23\)](#)
- [Etapa 5: testar a integração simulada \(p. 24\)](#)
- [Próximas etapas \(p. 25\)](#)

Etapa 1: criar a API

Nesta etapa, você cria uma API com um método GET e um parâmetro de consulta no console do API Gateway.

1. Se você ainda não tiver feito isso, conclua as etapas em [the section called “Pré-requisitos: prepare-se para compilar uma API no API Gateway” \(p. 11\)](#).
2. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
3. Se essa for a primeira vez que você usa o API Gateway, verá uma página que apresenta os recursos do serviço. Escolha Get Started. Quando o pop-up Create Example API (Criar API de exemplo) for exibido, escolha OK.

Se não for a primeira vez, escolha Create API (Criar API).



4. Em Choose the protocol (Escolher o protocolo), selecione REST.
5. Em Create new API (Criar nova API), selecione New API (Nova API).
6. Em Settings (Configurações):
 - Em API name (Nome da API), insira `my-api`.
 - Se desejar, digite uma descrição no campo Description (Descrição). Caso contrário, deixe-o vazio.
 - Deixe Endpoint Type (Tipo de endpoint) definido como Regional.
7. Escolha Create API.
8. No menu suspenso Actions (Ações), escolha Create Method (Criar método).
9. Sob o nome do recurso (/), você verá um menu suspenso. Escolha GET e depois escolha o ícone de marca de seleção para salvar sua opção.
10. Você verá um painel / – GET – Setup (/ – GET – Configuração). Em Integration Type (Tipo de integração), escolha Mock (Simulação).

/ - GET - Setup



Choose the integration point for your new method.

Integration type Lambda Function [i](#)

HTTP [i](#)

Mock [i](#)

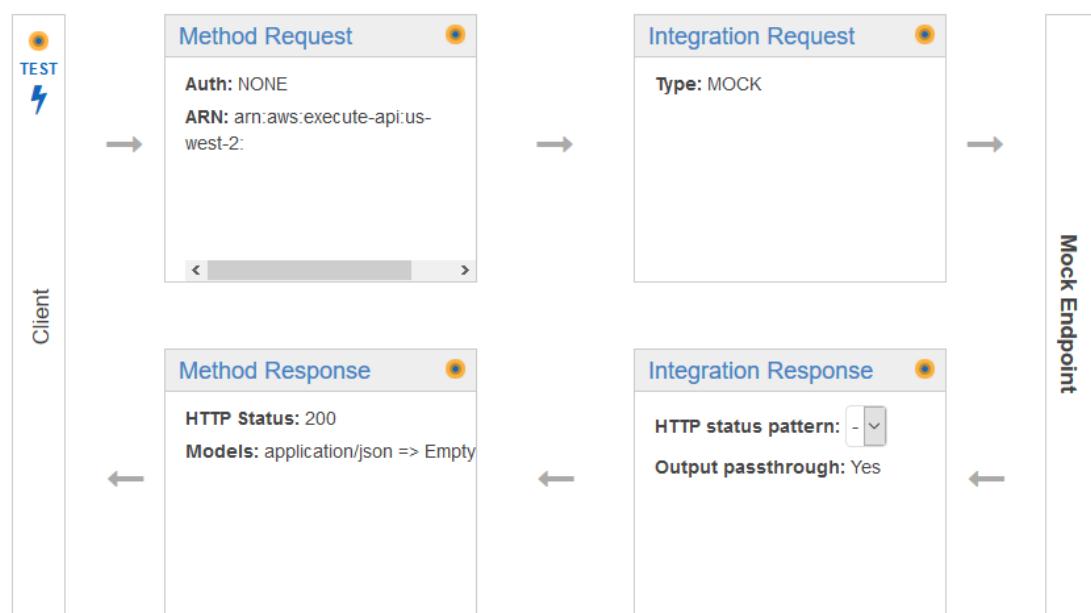
AWS Service [i](#)

VPC Link [i](#)

Save

11. Escolha Salvar. Agora, você verá um painel / – GET – Method Execution (/ – GET – Execução de método):

/ - GET - Method Execution



O painel Method Execution (Execução de método) contém estes itens, no sentido horário:

- Client (Cliente): essa caixa representa o cliente (navegador ou aplicativo) que chama o método GET da API. Se você escolher o link Test (Testar) e escolher Test (Testar), isso simulará uma solicitação GET de um cliente.
- Method Request (Solicitação de método): essa caixa representa a solicitação GET do cliente quando ela é recebida pelo API Gateway. Se você escolher Method Request (Solicitação de método), verá configurações para itens como autorização e modificação da solicitação de método antes que ela seja transmitida ao back-end como uma solicitação de integração.
- Integration Request (Solicitação de integração): essa caixa representa a solicitação GET que seria transmitida ao back-end. Para uma integração simulada, Mapping Templates (Modelos de

mapeamento) pode ser usado para especificar a resposta (o código e a mensagem de status HTTP) a ser retornada.

- Mock Endpoint (Endpoint simulado): essa caixa representa o back-end vazio.
- Integration Response (Resposta da integração): essa caixa representa a resposta do back-end, antes de ser transmitida ao cliente como uma resposta de método. Para uma integração simulada, a resposta é definida na caixa Integration Request (Solicitação de integração) (porque não há um back-end para fornecê-lo). Você pode configurar a resposta de integração para transformar uma saída arbitrária em um formato adequado para o aplicativo cliente. Você aprenderá como fazer isso mais adiante nessa demonstração.
- Method Response (Resposta de método): essa caixa representa os códigos de status HTTP que podem ser retornados ao cliente. Para uma integração simulada, o restante da resposta é configurado na caixa Integration Response (Resposta de integração) para cada código de status.

Para essa etapa, deixe tudo definido com os valores padrão.

Etapa 2: criar a integração simulada

Nessa etapa, em vez de integrar com um back-end, você adiciona um parâmetro de string de consulta ao método GET e especifica os códigos e mensagens de resposta que a API retornará.

Primeiro, crie o parâmetro de string de consulta:

1. No painel / - GET - Method Execution (/ - GET - Execução do método), selecione Method Request (Solicitação do método).
2. No painel / - GET - Method Request (/ - GET - Solicitação de método), expanda URL Query String Parameters (Parâmetros de string de consulta de URL).
3. Escolha Add query string (Adicionar string de consulta).
4. Digite `myParam` para o Name (Nome) e escolha o ícone de marca de seleção para salvar sua opção.

Depois, crie um modelo de mapeamento que mapeie os valores de parâmetro da string de consulta para os valores do código de status HTTP a serem retornados para o cliente

1. Escolha Method Execution (Execução de método).
2. Escolha Integration Request (Solicitação de integração).
3. Expanda Mapping Templates (Modelos de mapeamento).
4. Em Request body passthrough (Passagem do corpo da solicitação), escolha When there are no templates defined (recommended) (Quando não há modelos definidos (recomendado)).
5. Em Content-Type (Tipo de conteúdo), escolha application/json.
6. Substitua o conteúdo do modelo pelo seguinte:

```
{  
    #if( $input.params('myParam') == "myValue" )  
        "statusCode": 200  
    #else  
        "statusCode": 500  
    #end  
}
```

7. Escolha Salvar.

Etapa 3: definir a resposta bem-sucedida

Agora, você criará um modelo de mapeamento que mapeia o valor do código de status HTTP 200 para uma mensagem de êxito a ser retornada ao cliente.

1. Escolha Method Execution (Execução do método) e Integration Response (Resposta de integração).
2. Expanda a resposta 200 e a seção Mapping Templates (Modelos de mapeamento).
3. Em Content-Type (Tipo de conteúdo), escolha application/json.
4. No conteúdo do modelo, digite o seguinte:

```
{  
    "statusCode": 200,  
    "message": "Hello from API Gateway!"  
}
```

5. Escolha Salvar.

Note

Há dois botões Save (Salvar) nesse painel. Certifique-se de escolher um na seção Mapping Templates (Modelos de mapeamento).

Etapa 4: adicionar um código de status HTTP 500 e uma mensagem de erro

Nessa etapa, você adiciona um código de status HTTP 500 que o front-end pode retornar ao cliente e mapeia-o para uma mensagem de erro.

Primeiro, adicione o código de status 500:

1. Escolha Method Execution (Execução de método) e Method Response (Resposta de método).
2. Escolha Add response (Adicionar resposta).
3. Em HTTP status (status HTTP), digite 500 e escolha o ícone de marca de seleção para salvar a configuração.

Agora, crie um modelo de mapeamento que mapeie o valor do código de status 500 "retornado" para uma mensagem de erro do front-end para o cliente:

1. Escolha Method Execution (Execução do método) e Integration Response (Resposta de integração).
2. Escolha Add integration response (Adicionar resposta de integração).
3. Em HTTP status regex (Regex de status HTTP), insira `5\d{2}`.
4. Em Method response status (Status de resposta do método), escolha 500 no menu suspenso e selecione Save (Salvar).
5. Expanda a resposta 500 e a seção Mapping Templates (Modelos de mapeamento), caso ela ainda não esteja expandida.
6. Em Content-Type, escolha Add mapping template.
7. Na caixa em Content-Type (Tipo de conteúdo), insira `application/json` e escolha o ícone de marca de seleção para salvar sua opção.
8. No conteúdo do modelo, digite o seguinte:

```
{  
    "statusCode": 500,
```

```
        "message": "This is an error message."  
    }
```

9. Escolha Salvar.

Note

Há dois botões Save (Salvar) nesse painel. Certifique-se de escolher um na seção Mapping Templates (Modelos de mapeamento).

Etapa 5: testar a integração simulada

Nessa etapa, você testará a integração simulada.

1. Escolha Method Execution (Execução do método) e Test (Testar).
2. Em Query Strings (Strings de consulta), insira `myParam=myValue`.
3. Escolha Test.

O resultado deverá ser parecido com o seguinte:

Request: /?myParam=myValue



Status: 200

Latency: 5 ms

Response Body

```
{  
    "statusCode": 200,  
    "message": "Hello from API Gateway!"  
}
```

Response Headers

```
{"Content-Type": "application/json"}
```

Logs

```
Execution log for request 2083a122-7076-11e9-b3da-c5566e6284  
7c  
Tue May 07 03:13:46 UTC 2019 : Starting execution for reques  
t: 2083a122-7076-11e9-b3da-c5566e62847c  
Tue May 07 03:13:46 UTC 2019 : HTTP Method: GET, Resource Pa
```

4. Em Query Strings (Strings de consulta), insira `myParam=""`. Ou basta excluir tudo na caixa Query Strings (Strings de consulta).

5. Escolha Test.

O resultado deverá ser parecido com o seguinte:

Request: /?myParam=""

Status: 500

Latency: 5 ms

Response Body



```
{  
  "statusCode": 500,  
  "message": "This is an error message."  
}
```

Response Headers

```
{"Content-Type":"application/json"}
```

Logs

```
Execution log for request b7132dc8-7076-11e9-897f-832e244f10ba  
Tue May 07 03:17:59 UTC 2019 : Starting execution for request: b7132dc8-7076-11e9-897f-832e244f10ba  
Tue May 07 03:17:59 UTC 2019 : HTTP Method: GET, Resource Path: /
```

Próximas etapas

Explore um ou todos os tópicos a seguir para continuar a se familiarizar com o Amazon API Gateway.

Para saber mais a respeito	Acessar
Definição de solicitações e respostas de método	the section called “Configurar métodos de API REST” (p. 204)
Definição de respostas de integração	the section called “Configurar uma resposta de integração” (p. 226)
Configurar um nome de domínio personalizado para a API	the section called “Configurar um nome de domínio personalizado de API” (p. 676)

Para saber mais a respeito	Acessar
Ativar o CORS para a API	the section called “Habilitar o CORS para um recurso da API REST” (p. 423)

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

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

Vídeos do Amazon API Gateway

Essa seção fornece vários vídeos educativos do Amazon API Gateway da AWS que podem servir como recursos adicionais para você.

Tópicos

- [Vídeos do Amazon API Gateway da série de eventos Compilação na AWS do Twitch \(p. 26\)](#)
- [Outros vídeos do Amazon API Gateway \(p. 26\)](#)

Vídeos do Amazon API Gateway da série de eventos Compilação na AWS do Twitch

Os vídeos a seguir são da série de eventos [Compilação na AWS](#) no Twitch.tv.

[Introduction to Building Happy Little APIs \(Introdução à compilação de APIs boas e pequenas\) \(1 minuto, site do YouTube\)](#)

Esse vídeo apresenta brevemente a série de vídeos Building Happy Little APIs.

[Building Happy Little APIs | Episode 1 – I Didn’t Know Amazon API Gateway Did That \(Compilando APIs boas e pequenas | Episódio 1 – Eu não sabia que o Amazon API Gateway fazia isso\) \(60 minutos, site do YouTube\)](#)

Esse vídeo é uma introdução ao Amazon API Gateway e aos problemas que ele pode resolver. Ele aborda as partes móveis do API Gateway e fornece exemplos de possíveis casos de uso. Essa visão geral tem o objetivo de dar a você uma compreensão sólida do motivo pelo qual você deve usar o API Gateway e o que ele pode fazer por você.

[Building Happy Little APIs | Episode 2 – No REST for the Weary \(Compilando APIs boas e pequenas | Episódio 2 – Sem REST para o Weary\) \(55 minutos, site do YouTube\)](#)

Esse vídeo mostra como compilar um aplicativo simples que usa o Amazon API Gateway com uma função do AWS Lambda para um back-end. Você aprenderá como usar o Modelo de aplicativo sem servidor da AWS (AWS SAM) para modelar a API e o aplicativo por trás dela. Você também aprenderá a diferença entre integrações e proxies e quando usar cada um deles.

Outros vídeos do Amazon API Gateway

Os vídeos a seguir estão disponíveis no canal [AWS Online Tech Talks](#) no site do YouTube.

[Building APIs with Amazon API Gateway \(Compilando APIs com o Amazon API Gateway\)](#) (43 minutos, site do YouTube)

Esse vídeo descreve os recursos do Amazon API Gateway e mostra como você pode começar a compilar APIs.

[Best Practices for Building Enterprise Grade APIs with Amazon API Gateway \(Práticas recomendadas para compilar APIs de nível empresarial com o Amazon API Gateway\)](#) (40 minutos, site do YouTube)

Esse vídeo mostra como usar o Amazon API Gateway com outros serviços da AWS para projetar e operar APIs prontas para uso empresarial. Você aprenderá sobre práticas recomendadas para ajudar a criar, manter e proteger suas APIs corporativas.

Tutoriais do Amazon API Gateway

Os seguintes tutoriais fornecem exercícios práticos para ajudá-lo a saber mais sobre o API Gateway.

Tópicos

- [Criar uma API do API Gateway com integração à Lambda \(p. 28\)](#)
- [TUTORIAL: Crie uma API REST importando um exemplo \(p. 45\)](#)
- [Criar uma API do API Gateway com integração do HTTP \(p. 54\)](#)
- [TUTORIAL: Criar uma API com integração privada do API Gateway \(p. 89\)](#)
- [TUTORIAL: Criar uma API do API Gateway com integração da AWS \(p. 91\)](#)
- [TUTORIAL: Criar uma API REST Calc com duas integrações de serviços da AWS e uma integração não proxy do Lambda \(p. 96\)](#)
- [TUTORIAL: Criar uma API REST como um proxy do Amazon S3 no API Gateway \(p. 116\)](#)
- [TUTORIAL: Criar uma API REST como um proxy do Amazon Kinesis no API Gateway \(p. 142\)](#)

Criar uma API do API Gateway com integração à Lambda

Para criar uma API com integrações do Lambda, você pode usar uma integração de proxy do Lambda ou uma integração não proxy do Lambda.

Na integração de proxy do Lambda, a entrada da função do Lambda integrada pode ser expressa como qualquer combinação de cabeçalhos de solicitação, variáveis de caminho, parâmetros de strings de consulta e corpo. Além disso, a função Lambda pode usar definições de configuração da API para influenciar sua lógica de execução. Para um desenvolvedor de APIs, a configuração de integração de proxy Lambda é simples. Além de escolher uma determinada função do Lambda em uma determinada região, não há muito a se fazer. O API Gateway configura a solicitação de integração e a resposta de integração para você. Após a configuração, o método de API integrada pode evoluir com o back-end sem modificar as configurações existentes. Isso é possível porque o desenvolvedor da função Lambda de back-end analisa os dados da solicitação recebida e responde ao cliente com os resultados desejados quando nada está errado ou responde com mensagens de erro quando há algum erro.

Na integração não proxy do Lambda, é necessário garantir que a entrada da função do Lambda seja fornecida como a carga da solicitação de integração. Isso significa que você, como desenvolvedor de APIs, deve mapear quaisquer dados de entrada fornecidos pelo cliente como parâmetros da solicitação no corpo da solicitação de integração apropriada. Você também pode precisar traduzir o corpo da solicitação fornecida pelo cliente em um formato reconhecido pela função Lambda.

Tópicos

- [TUTORIAL: Criar uma API Hello World com integração de proxy do Lambda \(p. 29\)](#)
- [TUTORIAL: Criar uma API do API Gateway com integração de proxy do Lambda entre contas \(p. 34\)](#)
- [TUTORIAL: Criar uma API do API Gateway com integração não proxy do Lambda \(p. 36\)](#)

TUTORIAL: Criar uma API Hello World com integração de proxy do Lambda

A [integração de proxy do Lambda](#) (p. 227) é um tipo de integração da API do API Gateway leve e flexível que permite integrar um método de API ou uma API inteira a uma função do Lambda. A função do Lambda pode ser gravada 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:

- Crie uma função do Lambda "Hello, World!" que seja o back-end para a API.
- Criar e testar um "Hello, World!" API com integração de proxy do Lambda.

Tópicos

- [Criar um "Hello, World!" Lambda Função](#) (p. 29)
- [Criar um "Hello, World!" API](#) (p. 30)
- [Implantar e testar a API](#) (p. 32)

Criar um "Hello, World!" Lambda Função

Essa função retorna uma saudação ao chamador como um objeto JSON no seguinte formato:

```
{  
    "greeting": "Good {time}, {name} of {city}.[ Happy {day}! ]"  
}
```

Criar uma função do Lambda "Hello, World!" no console do Lambda

1. Faça login no console do Lambda em <https://console.aws.amazon.com/lambda>.
2. Na barra de navegação da AWS, selecione uma [região](#) (por exemplo, Leste dos EUA (Norte da Virgínia)).

Note

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

3. Selecione Functions (Funções) no painel de navegação.
4. Selecione Create function (Criar função).
5. Escolha Author from scratch.
6. Em Basic information (Informações básicas), faça o seguinte:
 - a. Em Function name (Nome da função), insira **GetStartedLambdaProxyIntegration**.
 - b. Na lista suspensa Runtime (Tempo de execução), selecione Node.js 8.10.
 - c. Em Permissions (Permissões), expanda Choose or create an execution role (Escolher ou criar uma função de execução). Na lista suspensa Execution role (Função de execução), selecione Create new role from AWS policy templates (Criar nova função de modelos de política da AWS).
 - d. Em Role name (Nome da função), insira **GetStartedLambdaBasicExecutionRole**.
 - e. Deixe em branco o campo Policy templates (Modelos de política).
 - f. Selecione Create function (Criar função).
7. Em Function code (Código de função), no editor de código em linha, copie/cole o seguinte código:

```
'use strict';
console.log('Loading hello world function');

exports.handler = async (event) => {
    let name = "you";
    let city = 'World';
    let time = 'day';
    let day = '';
    let responseCode = 200;
    console.log("request: " + JSON.stringify(event));

    if (event.queryStringParameters && event.queryStringParameters.name) {
        console.log("Received name: " + event.queryStringParameters.name);
        name = event.queryStringParameters.name;
    }

    if (event.queryStringParameters && event.queryStringParameters.city) {
        console.log("Received city: " + event.queryStringParameters.city);
        city = event.queryStringParameters.city;
    }

    if (event.headers && event.headers['day']) {
        console.log("Received day: " + event.headers.day);
        day = event.headers.day;
    }

    if (event.body) {
        let body = JSON.parse(event.body)
        if (body.time)
            time = body.time;
    }

    let greeting = `Good ${time}, ${name} of ${city}.`;
    if (day) greeting += ` Happy ${day}!`;

    let responseBody = {
        message: greeting,
        input: event
    };

    // The output from a Lambda proxy integration must be
    // in the following JSON object. The 'headers' property
    // is for custom response headers in addition to standard
    // ones. The 'body' property must be a JSON string. For
    // base64-encoded payload, you must also set the 'isBase64Encoded'
    // property to 'true'.
    let response = {
        statusCode: responseCode,
        headers: {
            "x-custom-header" : "my custom header value"
        },
        body: JSON.stringify(responseBody)
    };
    console.log("response: " + JSON.stringify(response))
    return response;
};
```

8. Escolha Salvar.

Criar um "Hello, World!" API

Agora, crie uma API para sua função do Lambda "Hello, World!" usando o console do API Gateway.

Criar um "Hello, World!" API

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Se essa for a primeira vez que você usa o API Gateway, verá uma página que apresenta os recursos do serviço. Escolha Get Started. 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).

3. Crie uma API vazia da seguinte maneira:
 - a. Em Choose the protocol (Escolher o protocolo), selecione REST.
 - b. Em Create new API (Criar nova API), selecione New API (Nova API).
 - c. Em Settings (Configurações):
 - Em API name (Nome da API), insira **LambdaSimpleProxy**.
 - Se desejar, digite uma descrição no campo Description (Descrição). Caso contrário, deixe-o vazio.
 - Deixe Endpoint Type (Tipo de endpoint) definido como Regional.
 - d. Escolha Create API (Criar API).
4. Crie o recurso `helloworld` da seguinte forma:
 - a. Selecione o recurso raiz (/) na árvore Resources (Recursos).
 - b. Selecione Create Resource (Criar recurso) no menu suspenso Actions (Ações).
 - c. Deixe Configure as proxy resource (Configurar como recurso de proxy) desmarcado.
 - d. Em Resource Name (Nome do recurso), insira **helloworld**.
 - e. Deixe Resource Path (Caminho do recurso) definido como /helloworld.
 - f. Deixe Enable API Gateway CORS (Habilitar CORS do API Gateway) desmarcado.
 - g. Escolha Create Resource (Criar recurso).
5. 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.

Para configurar o método ANY, faça o seguinte:

- a. Na lista Resources (Recursos), selecione /helloworld.
- b. No menu Actions (Ações), selecione Create method (Criar método).
- c. Selecione ANY no menu suspenso e selecione o ícone de marca de seleção.
- d. Deixe Integration type (Tipo de integração) definido como Lambda Function (Função do Lambda).
- e. Selecione Use Lambda Proxy Integration (Usar a integração de proxy do Lambda).
- f. No menu suspenso Lambda Region (Região do Lambda), selecione a região na qual você criou a função do Lambda `GetStartedLambdaProxyIntegration`.
- g. No campo Lambda Function (Função do Lambda), digite qualquer caractere e selecione `GetStartedLambdaProxyIntegration` no menu suspenso.
- h. Deixe Use Default Timeout (Usar o tempo limite padrão) marcado.
- i. Escolha Salvar.
- j. Selecione OK quando solicitado com Add Permission to Lambda Function (Adicionar permissão à função do Lambda).

Implantar e testar a API

Implantar a API no console do API Gateway

1. Escolha Deploy API (Implantar API) no menu suspenso Actions (Ações).
2. Em Deployment stage (Estágio de implantação), selecione [new stage] ([novo estágio]).
3. Em Stage name (Nome do estágio), insira **test**.
4. Se desejar, digite uma Stage description (Descrição do estágio).
5. Se desejar, digite uma Deployment description (Descrição da implantação).
6. Selecione Deploy (Implantar).
7. Anote o Invoke URL (URL de invocação) da API.

Uso do 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 digitar o URL do recurso helloworld da API em uma barra de endereço do navegador. Por exemplo:
<https://r275xc9bmd.execute-api.us-east-1.amazonaws.com/test/helloworld?name=John&city=Seattle>

Para outros métodos, você deve usar instalações de teste de API REST mais avançadas, tais como [POSTMAN](#) ou [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 seguinte comando cURL e cole-o na janela do terminal, substituindo **r275xc9bmd** pelo ID de API da sua API e **us-east-1** pela região em que a API foi implantada.

```
curl -v -X POST \
  'https://r275xc9bmd.execute-api.us-east-1.amazonaws.com/test/helloworld?
  name=John&city=Seattle' \
  -H 'content-type: application/json' \
  -H 'day: Thursday' \
  -d '{ "time": "evening" }'
```

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?name=John&city=Seattle" -H "content-type: application/json" -H "day:
  Thursday" -d "{ \"time\": \"evening\" }"
```

Você receberá uma resposta bem-sucedida com uma carga semelhante ao seguinte:

```
{
  "message": "Good evening, John of Seattle. Happy Thursday!",
  "input": {
    "resource": "/helloworld",
    "path": "/helloworld",
```

```
    "httpMethod": "POST",
    "headers": {"Accept": "*/*",
    "content-type": "application/json",
    "day": "Thursday",
    "Host": "r275xc9bmd.execute-api.us-east-1.amazonaws.com",
    "User-Agent": "curl/7.64.0",
    "X-Amzn-Trace-Id": "Root=1-1a2b3c4d-a1b2c3d4e5f6a1b2c3d4e5f6",
    "X-Forwarded-For": "72.21.198.64",
    "X-Forwarded-Port": "443",
    "X-Forwarded-Proto": "https"},
    "multiValueHeaders": {"Accept": ["*/*"],
    "content-type": ["application/json"],
    "day": ["Thursday"],
    "Host": ["r275xc9bmd.execute-api.us-east-1.amazonaws.com"],
    "User-Agent": ["curl/0.0.0"],
    "X-Amzn-Trace-Id": ["Root=1-1a2b3c4d-a1b2c3d4e5f6a1b2c3d4e5f6"],
    "X-Forwarded-For": ["11.22.33.44"],
    "X-Forwarded-Port": ["443"],
    "X-Forwarded-Proto": ["https"]},
    "queryStringParameters": {"city": "Seattle",
    "name": "John"
},
    "multiValueQueryStringParameters": {
        "city": ["Seattle"],
        "name": ["John"]
},
    "pathParameters": null,
    "stageVariables": null,
    "requestContext": {
        "resourceId": "3htbry",
        "resourcePath": "/helloworld",
        "http": {
            "Connection": "#0 to host r275xc9bmd.execute-api.us-east-1.amazonaws.com left intact"
        },
        "method": "POST",
        "extendedRequestId": "a1b2c3d4e5f6g7h=",
        "requestTime": "20/Mar/2019:20:38:30 +0000",
        "path": "/test/helloworld",
        "accountId": "123456789012",
        "protocol": "HTTP/1.1",
        "stage": "test",
        "domainPrefix": "r275xc9bmd",
        "requestTimeEpoch": 1553114310423,
        "requestId": "test-invoke-request",
        "identity": {
            "cognitoIdentityPoolId": null,
            "accountId": null,
            "cognitoIdentityId": null,
            "caller": null,
            "sourceIp": "test-invoke-source-ip",
            "accessKey": null,
            "cognitoAuthenticationType": null,
            "cognitoAuthenticationProvider": null,
            "userArn": null,
            "userAgent": "curl/0.0.0",
            "user": null
        }
    },
    "domainName": "r275xc9bmd.execute-api.us-east-1.amazonaws.com",
    "apiId": "r275xc9bmd"
},
    "body": "{ \"time\": \"evening\" }",
    "isBase64Encoded": false
}
```

Se você alterar POST para PUT na solicitação de método anterior, deverá obter a mesma resposta.

Para testar o método GET, copie o seguinte comando cURL e cole-o na janela do terminal, substituindo **r275xc9bmd** pelo ID da API da sua API e **us-east-1** pela região em que a API foi implantada.

```
curl -X GET \
'https://r275xc9bmd.execute-api.us-east-1.amazonaws.com/test/helloworld?Seattle?
name=John' \
-H 'content-type: application/json' \
-H 'day: Thursday'
```

Você deve obter uma resposta semelhante ao resultado da solicitação POST anterior, exceto que a solicitação GET não tem nenhuma carga. Então, o parâmetro body será null.

TUTORIAL: Criar uma API do API Gateway com integração de proxy do Lambda entre contas

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

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

Primeiro, criaremos a API de exemplo de [the section called “TUTORIAL: Crie uma API REST importando um exemplo” \(p. 45\)](#) em uma conta. Em seguida, criaremos uma função Lambda em outra conta. Por fim, usaremos a integração do Lambda entre contas para permitir que a API de exemplo use a função do Lambda criada na segunda conta.

Criar API para integração do Lambda entre contas do API Gateway

Primeiro, você criará a API de exemplo, conforme descrito em [the section called “TUTORIAL: Crie uma API REST importando um exemplo” \(p. 45\)](#).

Para criar a API de exemplo

1. Faça login no console do API Gateway.
2. Escolha Create API (Criar API) na página inicial de APIs do API Gateway.
3. Em Create new API (Criar nova API), escolha Examples API (Exemplos de API).
4. Em Endpoint Type (Tipo de endpoint), escolha Edge optimized (Otimizado para fronteiras).
5. Escolha Import (Importar) para criar a API de exemplo.

Criar a função Lambda de integração em outra conta

Agora você criará uma função Lambda em uma conta diferente daquela em que criou a API de exemplo.

Criar uma função Lambda em outra conta

1. Faça logon no console do Lambda usando uma conta diferente daquela em que você criou sua API do API Gateway.
2. Escolha Create function.
3. Escolha Author from scratch.
4. Em Author from scratch (Criar do zero), faça o seguinte:

- a. No campo de entrada Name (Nome), digite um nome de função.
- b. Na lista suspensa Runtime (Tempo de execução), selecione um tempo de execução com suporte. Neste exemplo, usamos Node.js 8.10.
- c. Na lista suspensa Role (Função), selecione Choose an existing role (Selecionar uma função existente), Create new role from template(s) (Criar nova função a partir de modelo(s)) ou Create a custom role (Criar uma função personalizada). Em seguida, siga as instruções posteriores para a opção.
- d. Escolha Create function (Criar função) para continuar.

Para este exemplo, ignoraremos a seção Designer e iremos para a próxima seção Function code (Código da função).

5. Role para baixo, até o painel Function code (Código da função).
6. Copie e cole a implementação da função Node.js de [the section called “TUTORIAL: API Hello World com integração de proxy do Lambda” \(p. 29\)](#).
7. No menu suspenso Runtime (Tempo de execução), selecione Node.js 8.10.
8. Escolha Salvar.
9. Anote o ARN completo da sua função (no canto superior direito do painel de funções 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 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.

Configurar sua integração do Lambda entre contas

1. No console do API Gateway, escolha sua API.
2. Selecione Resources (Recursos).
3. No painel Resources (Recursos), escolha o método GET de nível superior.
4. No painel Method Execution (Execução de método), escolha Solicitação de integração (Integration Request).
5. Em Integration type (Tipo de integração), escolha Lambda Function (Função do Lambda).
6. Marque a opção Use Lambda Proxy integration (Usar integração de proxy do Lambda).
7. Deixe Lambda Region (Região do Lambda) definido como a região da sua conta.
8. Para Lambda Function (Função do Lambda), copie/cole o ARN completo para a função do Lambda criada na sua segunda conta e escolha a marca de seleção.
9. Você verá uma janela pop-up que diz Adicionar permissão à função do Lambda: você selecionou uma função do Lambda de outra conta. Verifique se você tem uma política de função adequada para essa função. Você pode fazer isso executando o seguinte comando da AWS CLI da conta **123456789012**, seguido por uma string de comando `aws lambda add-permission`.
10. Copie e cole a string de comando `aws lambda add-permission` em uma janela da AWS CLI configurada para sua segunda conta. Isso concederá acesso para sua primeira conta à função Lambda da sua segunda conta.
11. Na janela pop-up da etapa anterior no console do Lambda, escolha OK.

12. Para ver a política atualizada da sua função no console do Lambda,

- a. Escolha sua função de integração.
- b. No painel Designer, escolha o ícone da chave.

No painel Function policy (Política da função), você verá uma política `Allow` com uma cláusula `Condition` em que `AWS:SourceArn` é o ARN do método `GET` da sua API.

TUTORIAL: Criar uma API do API Gateway com integração não proxy do Lambda

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

Para facilitar o aprendizado, escolhemos uma função do Lambda simples com configuração mínima da API para orientá-lo através das 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 Calc com duas integrações de serviços da AWS e uma integração não proxy do Lambda \(p. 96\)](#).

Antes de criar a API, configure o back-end do Lambda criando uma função do Lambda no AWS Lambda, descrita a seguir.

Tópicos

- [Criar uma função do Lambda para a integração não proxy do Lambda \(p. 36\)](#)
- [Criar uma API com integração não proxy do Lambda \(p. 40\)](#)
- [Testar a chamada do método de API \(p. 42\)](#)
- [Implantar a API \(p. 43\)](#)
- [Testar a API em uma etapa de implantação \(p. 44\)](#)
- [Limpar \(p. 45\)](#)

Criar uma função do Lambda para a integração não proxy do Lambda

Note

A criação de funções do Lambda pode resultar em cobranças na sua 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 Node.js dessa função `GetStartedLambdaIntegration` do Lambda é conforme se segue:

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

```
console.log('Loading function');

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

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

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

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

Para a integração personalizada do Lambda, o API Gateway passa 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 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 chamador. A mensagem é no padrão "Good [morning|afternoon|day], [`name|you`] in [`city|World`]. Happy `day`!". Presume-se que a entrada para a função Lambda seja um dos seguintes objetos JSON:

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

Para obter mais informações, consulte o [AWS Lambda Developer Guide](#).

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 fluxos do CloudWatch e adicione entradas de log aos fluxos. 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 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 com permissões do IAM \(p. 378\)](#).

Em comparação à `GetStartedLambdaProxyIntegration`, a função do Lambda para a integração de proxy do Lambda, a função do Lambda `GetStartedLambdaIntegration` para a integração personalizada do Lambda aceita apenas entradas 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 qualquer dado 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 back-end. 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`.

Para 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. Siga um destes procedimentos:
 - Se a página de boas-vindas for exibida, escolha Get Started Now (Comece a usar agora) e, em seguida, Create a function (Criar uma função).
 - Se a página da lista Lambda > Functions (Lambda > Funções) for exibida, escolha Create a function (Criar uma função).
3. Escolha Author from scratch.
4. Na tela Author from scratch (Criar do zero), faça o seguinte:
 - a. Em Name (Nome), digite `GetStartedLambdaIntegration` como o nome da função do Lambda.
 - b. Em Runtime (Tempo de execução), selecione Node.js 8.10.
 - c. Em Role (Função), selecione Create new role from template(s).
 - d. Para Role Name (Nome da função), digite um nome para sua função (por exemplo, `GetStartedLambdaIntegrationRole`).
 - e. Em Policy templates (Modelos de políticas), escolha Simple Microservice permissions.
 - f. Escolha Create function.
5. No painel Configure function (Configurar função), em Function code (Código de função) faça o seguinte:
 - a. Escolha Edit code inline (Editar código em linha), caso ele ainda não seja exibido, em Content entry type (Tipo de entrada de conteúdo).
 - b. Deixe Handler definido como `index.handler`.
 - c. Defina Runtime (Tempo de execução) como Node.js 8.10.
 - d. Copie o código da função do Lambda listado no início desta seção e cole-o no editor de código em linha.
 - e. Deixe as opções padrão para todos os outros campos nesta seção.
 - f. Escolha Salvar.
6. Para testar a função recém-criada, escolha Configure test events (Configurar eventos de teste) em Select a test event... (Selecionar um evento de teste...).
 - a. Em Create new event (Criar novo evento), substitua as instruções de código padrão pelas seguintes, digite `HelloWorldTest` para o nome do evento e escolha Create (Criar).

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

- b. Escolha Test (Testar) para invocar a função. A seção Execution result: succeeded (Resultada da execução: bem-sucedida) é exibida. Expanda Detail (Detalhe) e você visualizará a saída a seguir.

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

A saída também é gravada em logs do CloudWatch.

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 Lambda. Ela permite a chamada `CreateLogGroup` do CloudWatch para qualquer recurso do CloudWatch de sua conta na região onde a função do Lambda é criada. Esta política também permite a criação de fluxos e eventos de registro em log do CloudWatch para a função `HelloWorldForLambdaIntegration` do Lambda.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "logs:CreateLogGroup",  
            "Resource": "arn:aws:logs:region:account-id:*"  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "logs:CreateLogStream",  
                "logs:PutLogEvents"  
            ],  
            "Resource": [  
                "arn:aws:logs:region:account-id:log-group:/aws/lambda/  
GetStartedLambdaIntegration:*"  
            ]  
        }  
    ]  
}
```

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 a `lambda.amazonaws.com`. Esta é a relação de confiança:

```
{  
    "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 em log no CloudWatch Logs.

Se você não usou o Console de gerenciamento da AWS para criar a função do Lambda, deve seguir esses exemplos para criar a função e as políticas do IAM necessárias e, em seguida, anexar a função manualmente à sua função.

Criar uma API com integração não proxy do Lambda

Com a função Lambda (`GetStartedLambdaIntegration`) criada e testada, você está pronto para expor a função através de uma API do API Gateway. Para fins de ilustração, a função 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 adequadamente. 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 back-end do Lambda.

Para criar uma API com integração personalizada do Lambda com uma função do Lambda

1. Inicie o console do API Gateway.
2. Selecione Create new API (Criar nova API).
 - a. Digite `GetStartedLambdaNonProxyIntegration` em API name (Nome da API).
 - b. Digite uma descrição da API em Description (Descrição) ou deixe o campo em branco.
 - c. Escolha Create API (Criar API).
3. Escolha o recurso raiz (/) em Resources (Recursos). No menu Actions (Ações), escolha Create Resource (Criar recurso).
 - a. Digite `city` para Resource Name (Nome do recurso).
 - b. Substitua Resource Path (Caminho do recurso) por `{city}`. Este é um exemplo da variável de caminho do modelo usada para receber entradas do cliente. Posteriormente, vamos mostrar como mapear essa variável de caminho para a entrada da função Lambda usando um modelo de mapeamento.
 - c. Selecione a opção Enable API Gateway Cors (Habilitar CORS do API Gateway).
 - d. Escolha Create Resource (Criar recurso).
4. Com o recurso `/ {city}` recém-criado em destaque, escolha Create Method (Criar método) em Actions (Ações).
 - a. Escolha ANY no menu suspenso de métodos HTTP. 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.
 - b. Para salvar a configuração, escolha a marca de seleção.
5. Em Method Execution (Execução de método), faça o seguinte para o método `/ {city} ANY`:
 - a. Escolha Lambda Function (Função do Lambda) para Integration type.
 - b. Deixe a caixa Use Lambda Proxy integration (Usar integração de proxy do Lambda) desmarcada.
 - c. Escolha a região onde você criou a função do Lambda; por exemplo, `us-west-2`.
 - d. Digite o nome da sua função do Lambda em Lambda Function (Função do Lambda), por exemplo, `GetStartedLambdaIntegration`.
 - e. Deixe a caixa Use Default timeout (Usar o tempo limite padrão) marcada.
 - f. Escolha Salvar.
 - g. Escolha OK no popup Add Permission to Lambda Function (Adicionar permissão à função do Lambda) para que o API Gateway configure as permissões de acesso necessárias para a API invocar a função do Lambda integrada.

6. Nesta etapa, você configurará o seguinte:

- Um parâmetro de string de consulta (`time`)
- Um parâmetro de cabeçalho (`day`)
- Uma propriedade de carga (`callerName`)

Na ocasião da execução, o cliente pode usar esses parâmetros de solicitação e o corpo da solicitação para fornecer a hora do dia, o dia da semana e o nome do chamador. Você já configurou a variável de caminho `{city}`.

- a. No painel Method Execution (Execução do método), selecione Method Request (Solicitação do método).
- b. Expanda a seção URL Query String Parameters (Parâmetros da string de consulta de URL). Escolha Add query string (Adicionar string de consulta). Digite `time` em Name (Nome). Selecione a opção Required (Obrigatório) e escolha o ícone de marca de seleção para salvar a configuração. Deixe Caching (Armazenamento em cache) em branco para evitar uma cobrança desnecessária para este exercício.
- c. Expanda a seção HTTP Request Headers (Cabeçalhos de solicitação HTTP). Escolha Add header (Adicionar cabeçalho). Digite `day` em Name (Nome). Selecione a opção Required (Obrigatório) e escolha o ícone de marca de seleção para salvar a configuração. Deixe Caching (Armazenamento em cache) em branco para evitar uma cobrança desnecessária para este exercício.
- d. Para definir a carga do método de solicitação, faça o seguinte:
 - i. Para definir um modelo, escolha Models (Modelos) na API do painel de navegação principal do API Gateway e, em seguida, escolha Create (Criar).
 - ii. Digite `GetStartedLambdaIntegrationUserInput` para Model name (Nome do modelo).
 - iii. Digite `application/json` para Content type (Tipo de conteúdo).
 - iv. Deixe Model description (Descrição do modelo) em branco.
 - v. Copie a seguinte definição de esquema no editor Model schema (Esquema do modelo):

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "GetStartedLambdaIntegrationInputModel",
  "type": "object",
  "properties": {
    "callerName": { "type": "string" }
  }
}
```

- vi. Escolha Create model (Criar modelo) para concluir a definição do modelo de entrada.
- vii. Escolha Resources (Recursos), selecione o método `{city} ANY`, escolha Method Request (Solicitação de método) e expanda Request body (Corpo da solicitação). Escolha Add model (Adicionar modelo). Digite `application/json` para Content type (Tipo de conteúdo). Escolha `GetStartedLambdaIntegrationInput` para Model name (Nome do modelo). Escolha o ícone de marca de seleção para salvar a configuração.
7. Escolha o método `{city} ANY` e selecione Integration Request (Solicitação de integração) para configurar um modelo de mapeamento do corpo. Nesta etapa, você mapeará o parâmetro de solicitação do método previamente configurado de `nameQuery` ou `nameHeader` para a carga JSON, conforme exigido pela função do Lambda de back-end.
 - a. Expanda a seção Mapping Templates (Modelos de mapeamento). Escolha Add mapping template (Adicionar modelo de mapeamento). Digite `application/json` para Content-Type (Tipo de conteúdo). Escolha o ícone de marca de seleção para salvar a configuração.
 - b. No pop-up exibido, escolha Yes, secure this integration (Sim, garantir essa integração).

- c. Consulte a recomendação When there are no templates defined para Request body passthrough (Passagem do corpo da solicitação).
- d. Escolha GetStartedLambdaIntegrationUserInput em Generate template (Gerar modelo) para gerar um modelo de mapeamento inicial. Essa opção está disponível porque você definiu um esquema de modelo, sem o qual seria necessário gravar o modelo de mapeamento do zero.
- e. Substitua o script de mapeamento gerado no editor de modelo de mapeamento pelo seguinte:

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

- f. Escolha Salvar.

Testar a chamada do método de API

O console do API Gateway fornece uma instalação de testes para que você teste a chamada 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.

Para testar a invocação do método ANY /{city}

1. Em Method Execution (Execução do método), escolha Test (Testar).
2. Escolha POST na lista suspensa Method (Método).
3. Em Path (Caminho), digite **Seattle**.
4. Em Query Strings (Strings de consulta), digite **time=morning**.
5. Em Headers (Cabeçalhos), digite **day:Wednesday**.
6. Em Request Body (Corpo da solicitação), digite **{ "callerName": "John" }**.
7. Escolha Test.
8. Verifique se a carga de resposta retornada é como se segue:

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

9. Você também pode visualizar os logs para examinar como o API Gateway processa a solicitação e 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=FOoDYXdzELL//////////wEaDMHGzEdEOT/VvGhabiK3AzgKrJw
  +3zLqJZG4PhOq12K6W21+QotY2rrZyOzqghLoiuRg3CAYNQ2eqgL5D54+63ey9bIdtwHGo
  yBdq8ecWxJK/YnT2Rau0L9HCG5p7FC05h3Ivw1FfvcidQNXeYvsKJTLXI05/
  yEnY3ttiAwpNYLOezD9Es8rBfyruHfJf0qextKlsC8DymCcqlGkig8qLKCZ0hWJWWwiP
  JfG17laabXs
  ++ZhCa4hdZo4iqlG729DE4gaV1mJVdoAagIUwLMo+y4NxFDu0r710/
  EO5nYcCrppGVVBYiGk7H4T6sXuhTkbNNqVmXtV3ch5b0lh7 [TRUNCATED]
Thu Aug 31 01:07:25 UTC 2017 : Endpoint request body after transformations: {
  "city": "Seattle",
  "time": "morning",
  "day": "Wednesday",
  "name" : "John"
}
Thu Aug 31 01:07:25 UTC 2017 : Sending request to https://lambda.us-
west-2.amazonaws.com/2015-03-31/functions/arn:aws:lambda:us-
west-2:123456789012:function:GetStartedLambdaIntegration/invocations
Thu Aug 31 01:07:25 UTC 2017 : Received response. Integration latency: 328 ms
Thu Aug 31 01:07:25 UTC 2017 : Endpoint response body before transformations:
  {"greeting":"Good morning, John of Seattle. Happy Wednesday!"}
Thu Aug 31 01:07:25 UTC 2017 : Endpoint response headers: {x-amzn-Remapped-Content-
Length=0, x-amzn-RequestId=c0475a28-8de8-11e7-8d3f-4183da788f0f, Connection=keep-
alive, Content-Length=62, Date=Thu, 31 Aug 2017 01:07:25 GMT, X-Amzn-Trace-
Id=root=1-59a7614d-373151b01b0713127e646635;sampled=0, Content-Type=application/json}
Thu Aug 31 01:07:25 UTC 2017 : Method response body after transformations:
  {"greeting":"Good morning, John of Seattle. Happy Wednesday!"}
Thu Aug 31 01:07:25 UTC 2017 : Method response headers: {X-Amzn-Trace-
Id=sampled=0;root=1-59a7614d-373151b01b0713127e646635, Content-Type=application/json}
Thu Aug 31 01:07:25 UTC 2017 : Successfully completed execution
Thu Aug 31 01:07:25 UTC 2017 : Method completed with status: 200
```

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 a API no painel APIs ou escolha um recurso ou método no painel Resources (Recursos). Escolha Deploy API (Implantar API) no menu suspenso Actions (Ações).

2. Em Deployment stage (Estágio de implantação), escolha New Stage (Novo estágio).
3. Em Stage name (Nome do estágio), digite um nome, por exemplo, **test**.

Note

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

4. Em Stage description (Descrição do estágio), digite uma descrição ou deixe em branco.
5. Em Deployment description (Descrição da implantação), digite uma descrição ou deixe em branco.
6. Escolha Deploy (Implantar). Após a implantação bem-sucedida da API, você verá o URL básico da API (o nome de host padrão mais o nome do estágio) exibidos como Invoke URL (Invocar URL) na parte superior do Stage Editor (Editor de estágio). O padrão geral deste URL básico 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 do caminho URL ou parâmetros de strings de consulta, você pode digitar o URL de recurso da API em um navegador. Para outros métodos, você deve usar instalações de teste de API REST mais avançadas, tais como [POSTMAN](#) ou [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.

3. Para testar GET /Boston?time=morning:

Copie o seguinte comando cURL e cole-o na janela do terminal.

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

Você obtém uma resposta 400 Bad Request com a seguinte mensagem de erro:

```
{"message": "Invalid request body"}
```

Isso ocorre porque a solicitação GET que você enviou não pode aceitar uma carga útil e falha na validação da solicitação.

Limpar

Se você não precisar mais das funções Lambda que criou para este passo-a-passo, poderá excluí-las agora. Você também pode excluir os recursos do IAM acompanhantes.

Warning

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

Se você excluir um recurso do IAM do qual uma função do Lambda depende, esta função do Lambda 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 IAM novamente, você deverá recriá-lo.

Para excluir as funções 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 GetHelloWorld, selecione Actions (Ações) e, em seguida, Delete function (Excluir função). Quando solicitado, escolha Delete (Excluir) novamente.
3. Na lista de funções, escolha GetHelloWithName, selecione Actions (Ações) e, em seguida, Delete function (Excluir função). Quando solicitado, escolha Delete (Excluir) novamente.

Para 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 APIGatewayLambdaExecRole, selecione Role Actions (Ações da função) e, em seguida, Delete Role (Excluir função). Quando solicitado, escolha Sim, excluir.
4. Em Details (Detalhes), escolha Policies (Políticas).
5. Na lista de políticas, escolha APIGatewayLambdaExecPolicy, selecione Policy Actions (Ações da política) e, em seguida, Delete (Excluir). Quando solicitado, escolha Delete (Excluir).

Você chegou ao final deste passo a passo.

TUTORIAL: Crie 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, você pode 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 back-end HTTP de `http://petstore-demo-endpoint.execute-api.com/petstore/pets`.

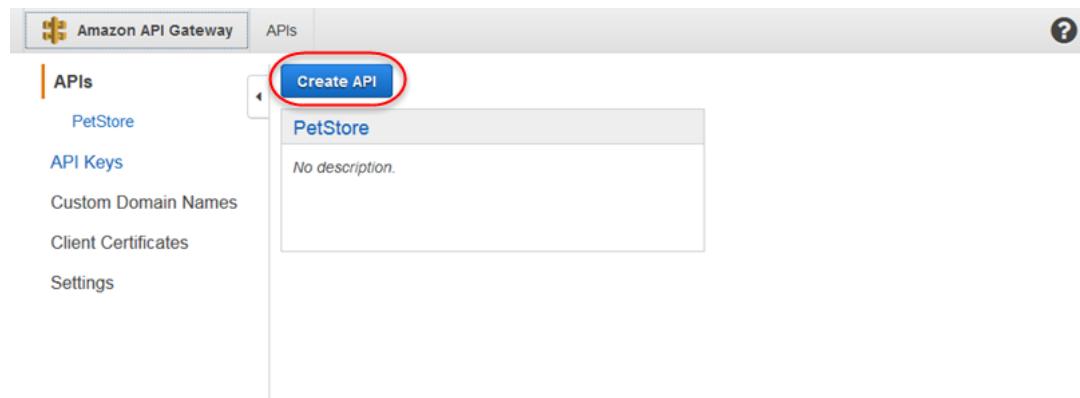
- `GET /`: para acesso de leitura ao recurso raiz da API que não está integrado a qualquer endpoint de back-end. O API Gateway responde com uma visão geral do site PetStore. Este é um exemplo do tipo de integração **MOCK**.
- `GET /pets`: para acesso de leitura ao recurso `/pets` da API que está integrado ao recurso de back-end `/pets` de nome idêntico. O back-end 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 back-end `/petstore/pets`. Depois de receber uma solicitação correta, o back-end adiciona o animal de estimação especificado à PetStore e retorna o resultado para o chamador. 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 back-end retorna o animal de estimação especificado encontrado na PetStore. O URL do endpoint HTTP de back-end 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. Se você ainda não tiver feito isso, conclua as etapas em [the section called “Pré-requisitos: prepare-se para compilar uma API no API Gateway” \(p. 11\)](#).
2. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
3. Faça uma das coisas a seguir:
 - a. Se esta é a primeira API em sua conta, escolha Get Started (Comece a usar) na página de boas-vindas do console do API Gateway.
Se solicitado com dicas, escolha OK para fechá-los e continuar.
 - b. Se essa não é sua primeira API, escolha Create API (Criar API) na página inicial de APIs do API Gateway:



4. Em Create new API (Criar nova API), escolha Example API (API de exemplo) e escolha Import (Importar) para criar a API de exemplo. Para a sua primeira API, o console do API Gateway começa com esta opção como padrão.

Create new API

In Amazon API Gateway, an API refers to a collection of resources and methods that can be invoked through HTTPS endpoints.

New API Clone from existing API Import from Swagger Example API

Example API

Learn about the service by importing an example API and turning on hints throughout the console.

```
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 with 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}  
24
```

Fail on warnings Ignore warnings

Import

Você pode percorrer a definição do OpenAPI para conhecer os detalhes dessa API de exemplo antes de escolher Import (Importar).

5. A API recém-criada é mostrada da seguinte forma:

The screenshot shows the 'Resources' section on the left with a tree view of API endpoints:

- / (Root)
 - GET
- /pets
 - GET
 - OPTIONS
 - POST
- /{petId}
 - GET
 - OPTIONS

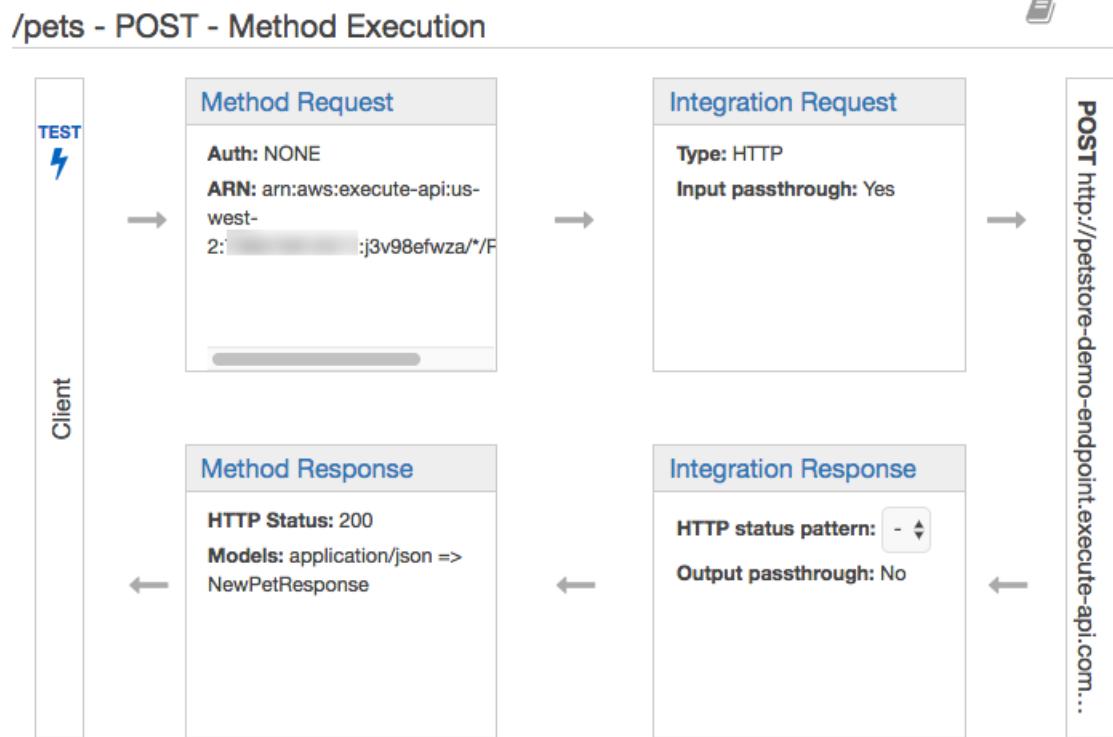
The 'Actions' dropdown is visible at the top. To the right, under '/ Methods', the 'GET' method for the root endpoint is selected, showing its configuration:

Mock Endpoint

Authorization: None
API Key: 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 seus métodos são listados no painel Methods (Métodos) à direita. Cada método exibe um breve resumo abaixo dele, incluindo sua URL de endpoint, o tipo de autorização e o requisito de Chave de API.

- 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:



O painel Method Execution (Execução de método) apresenta uma exibição lógica da estrutura e dos comportamentos do método escolhido (POST /pets): Method Request (Solicitação de método) e

Method Response (Resposta de método) são a interface da API com o front-end da API (um cliente), enquanto Integration Request (Solicitação de integração) e Integration Response (Resposta de integração) são interface da API com o back-end (`http://petstore-demo-endpoint.execute-api.com/petstore/pets`). Um cliente usa a API para acessar um recurso do back-end por meio de Method Request (Solicitação de método). O API Gateway traduz a solicitação do cliente, se necessário, para o formato aceito pelo back-end em Integration Request (Solicitação de integração) antes de encaminhar a solicitação para o back-end. A solicitação transformada é conhecida como a solicitação de integração. De maneira semelhante, o back-end retorna a resposta para o API Gateway em Integration Response (Resposta de integração). Em seguida, o API Gateway a direciona para a Method Response (Resposta de método) antes de enviar ao cliente. Novamente, se necessário, API Gateway pode mapear os dados da resposta de back-end 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 back-end real. A Integration Response (Resposta de integração) correspondente é configurada para retornar 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 Method Response (Resposta de método). Você pode usar a integração fictícia para testar uma API sem exigir um endpoint de back-end. 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 back-end 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. Abordamos a configuração do método em [TUTORIAL: Criar uma API com integração não proxy HTTP \(p. 59\)](#). Por enquanto, nos concentraremos em testar a API para fornecer uma experiência de usuário de ponta a ponta.

7. Escolha Test (Testar) mostrado em Client (Cliente) (como mostra a imagem anterior) para iniciar o teste. Por exemplo, para testar o método `POST /pets`, digite a seguinte carga útil `{ "type": "dog", "price": 249.99 }` no Request Body (Corpo da solicitação) antes de escolher o botão Test (Testar).

[← Method Execution](#) /pets - POST - Method Test

Make a test call to your method with the provided input

Path

No path parameters exist for this resource. You can define path parameters by using the syntax {myPathParam} in a resource path.

Query Strings

No query string parameters exist for this method. You can add them via Method Request.

Headers

No header parameters exist for this method. You can add them via Method Request.

Stage Variables

No stage variables exist for this method.

Client Certificate

No client certificates have been generated.

Request Body

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

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

8. Os resultados são exibidos da seguinte forma:

Request: /pets

Status: 200

Latency: 566 ms

Response Body

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

Response Headers

```
{"Access-Control-Allow-Origin":"*","X-Amzn-Trace-Id":"Root=1-59287e14-bd5f1d07c673367be0739eae","Content-Type":"application/json"}
```

Logs

```
Execution log for request test-request  
Fri May 26 19:12:20 UTC 2017 : Starting execution for request: test-invoke-request  
Fri May 26 19:12:20 UTC 2017 : HTTP Method: POST, Resource Path: /pets  
Fri May 26 19:12:20 UTC 2017 : Method request path: {}  
Fri May 26 19:12:20 UTC 2017 : Method request query string: {}  
Fri May 26 19:12:20 UTC 2017 : Method request headers: {}  
Fri May 26 19:12:20 UTC 2017 : Method request body before transformations: {"type": "dog", "price": 249.99}  
Fri May 26 19:12:20 UTC 2017 : Endpoint request URI: http://petstore-demo-endpoint.execute-api.com/petstore/pets  
Fri May 26 19:12:20 UTC 2017 : Endpoint request headers: {x-amzn-apigateway-api-id=4wk1k4onj3, Accept=application/json, User-Agent=AmazonAPIGateway_4wk1k4onj3, X-Amzn-Trace-Id=Root=1-59287e14-bd5f1d07c673367be0739eae, Content-Type=application/json}  
Fri May 26 19:12:20 UTC 2017 : Endpoint request body after transformations: {"type": "dog", "price": 249.99}  
Fri May 26 19:12:20 UTC 2017 : Endpoint response body before transformations:  
  "pet": {  
    "type": "dog",  
    "price": 249.99  
  },  
  "message": "success"  
}  
Fri May 26 19:12:20 UTC 2017 : Endpoint response headers: {Connection=keep-alive, Content-Length=81, Date=Fri, 26 May 2017 19:12:20 GMT, Content-Type=application/json; charset=utf-8, X-Powered-By=Express}  
Fri May 26 19:12:20 UTC 2017 : Method response body after transformations:  
  "pet": {  
    "type": "dog",  
    "price": 249.99  
  },  
  "message": "success"  
}  
Fri May 26 19:12:20 UTC 2017 : Method response headers: {Access-Control-Allow-Origin=*, X-Amzn-Trace-Id=Root=1-59287e14-bd5f1d07c673367be0739eae, Content-Type=application/json}  
Fri May 26 19:12:20 UTC 2017 : Successfully completed execution  
Fri May 26 19:12:20 UTC 2017 : Method completed with status: 200
```

A entrada Logs 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

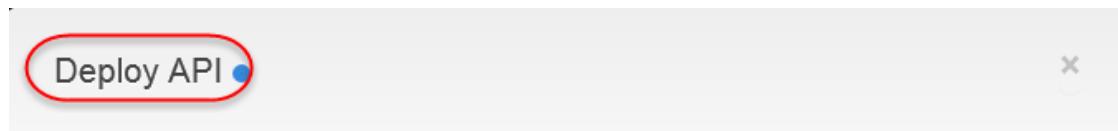
solicitação de integração para o back-end e, de maneira semelhante, a resposta do back-end é passada por meio da resposta da integração para a resposta do método.

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

9. Para implantar uma API de exemplo, selecione a API PetStore e depois escolha Deploy API (Implantar API) no menu Actions (Ações).

The screenshot shows the AWS API Gateway 'Actions' dropdown for the root resource '/'. The 'API ACTIONS' section contains five options: 'Deploy API', 'Import API', 'Edit API Documentation', and 'Delete API'. The 'Deploy API' option is circled in red.

Em Deploy API (Implantar API), para Deployment stage (Estágio de implantação), escolha [New Stage] ([Novo estágio]), pois essa é a primeira implantação da API. Digite um nome (por exemplo, **test**) em Stage name (Nome do estágio) e, opcionalmente, digite as descrições em Stage description (Descrição do estágio) e Deployment description (Descrição da implantação). Escolha Deploy (Implantar).



Choose a stage where your API will be deployed. For example, a test version of your API could be deployed to a stage named beta.

No painel Stage Editor (Editor de estágio) resultante, Invoke URL (Invocar URL) exibe a URL para invocar a solicitação de método GET / da API.

10. Em Stage Editor (Editor de estágio), siga o link Invoke URL (Invocar URL) para enviar a solicitação do método GET / em um navegador. Uma resposta bem-sucedida retornará o resultado, gerada a partir do modelo de mapeamento na resposta de integração.
11. No painel de navegação Stages (Estágios), expanda o estágio test (teste), selecione GET em / pets/{petId} e depois copie o valor Invoke URL (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 AWS_IAM fosse usada, você assinaria a solicitação usando os protocolos [Signature Version 4](#) (SigV4). Para obter um exemplo dessa solicitação, consulte [the section called "TUTORIAL: Criar uma API com integração não proxy HTTP" \(p. 59\)](#).

Criar uma API do API Gateway com integração do HTTP

Para criar uma API com integração HTTP, você pode usar uma integração de proxy HTTP ou uma integração HTTP personalizada. Recomendamos que você use a integração de proxy HTTP sempre que possível para a configuração simplificada da API, oferecendo recursos potentes e versáteis. A integração HTTP personalizada pode ser atraente se for necessário transformar os dados da solicitação do cliente para o back-end ou transformar os dados da resposta do back-end para o cliente.

Tópicos

- [TUTORIAL: Criar uma API com integração de proxy HTTP \(p. 54\)](#)
- [TUTORIAL: Criar uma API com integração não proxy HTTP \(p. 59\)](#)

TUTORIAL: Criar uma API com integração de proxy 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 passa a solicitação de método enviada pelo cliente para o back-end. 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 back-end 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 back-end interajam diretamente sem qualquer intervenção do API Gateway após a configuração do método da API, exceto por problemas conhecidos, como caracteres incompatíveis, que estão listados em [the section called "Observações importantes" \(p. 730\)](#).

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 back-end 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 através de um de recurso de proxy genérico {proxy+} e criamos o espaço reservado do método HTTP de ANY.

Tópicos

- [Criação de uma API com integração de proxy HTTP usando o console do API Gateway \(p. 54\)](#)
- [Testar uma API com a integração de proxy HTTP \(p. 56\)](#)

Criação de 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 back-end HTTP usando o console do API Gateway. O back-end HTTP é o site do PetStore (<http://petstore-demo-endpoint.execute-api.com/petstore/pets>) de [TUTORIAL: Criar uma API com](#)

integração não proxy HTTP (p. 59), no qual capturas de tela são usadas como auxílios visuais para ilustrar os elementos de UI do API Gateway. Se você é novo no uso do console do API Gateway para criar uma API, pode ser necessário seguir essa seção primeiro.

Para criar uma API com integração de proxy HTTP com o site PetStore por meio de um recurso de proxy.

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Para criar uma API, escolha Criar nova API (para criar a primeira API) ou Criar API (para criar uma API subsequente). Depois, faça o seguinte:
 - a. Escolha Nova API.
 - b. Digite um nome em Nome da API.
 - c. Opcionalmente, adicione uma breve descrição em Descrição.
 - d. Escolha Criar API.

Para este tutorial, use `ProxyResourceForPetStore` para o nome da API.

3. Para criar um recurso dependente, escolha um item do recurso pai na árvore Recursos e escolha Criar recurso no menu suspenso Ações. Depois, no painel New Child Resource, faça o seguinte:
 - a. Selecione a opção Configurar como recurso de proxy para criar um recurso de proxy. Caso contrário, mantenha a opção desmarcada.
 - b. Digite um nome no campo de texto Nome do recurso*.
 - c. Digite um novo nome ou use o nome padrão no campo de texto Caminho de recurso*.
 - d. Escolha Create Resource.
 - e. Selecione Enable API Gateway CORS, se necessário.

Para este tutorial, selecione Configure as proxy resource (Configurar como recurso de proxy). Em Resource Name (Nome do recurso), use o padrão, `proxy`. Em Resource Path (Caminho do recurso), use `/ {proxy+}`. Selecione Enable API Gateway CORS (Habilitar CORS do API Gateway).

The screenshot shows the 'New Child Resource' dialog in the AWS API Gateway console. The URL in the address bar is `> ProxyResourceForPetStore (miqyuu3lfg) > Resources > / (7ryftl47g0) > Create`. The dialog has tabs for 'Resources' and 'Actions'. The main area is titled 'New Child Resource' with the sub-instruction 'Use this page to create a new child resource for your resource.' Below this, there's a section for 'Configure as proxy resource' with a checked checkbox. The 'Resource Name*' field contains 'proxy'. The 'Resource Path*' field contains '/ {proxy+}'. A note explains that you can add path parameters using brackets like '{username}' and that '/{proxy+}' catches all requests to its sub-resources. At the bottom, there are 'Cancel' and 'Create Resource' buttons, with a note '* Required'.

4. Para configurar o método ANY para integração com o back-end HTTP, faça o seguinte:
 - a. Escolha o recurso recém-criado e escolha Create Method no menu suspenso Actions.

- b. Escolha ANY na lista suspensa do método HTTP e o ícone de marca de seleção para salvar a opção.
- c. Escolha HTTP Proxy como tipo de integração.
- d. Digite um URL de recursos de back-end HTTP em Endpoint URL.
- e. Use as configurações padrão para outros campos.
- f. Escolha Salvar para terminar a configuração do método ANY.

Para este tutorial, use `http://petstore-demo-endpoint.execute-api.com/{proxy}` para Endpoint URL (URL de endpoint).

/{{proxy+}} - ANY - Setup

API Gateway will configure your ANY method as a proxy integration. Proxy integrations can communicate with HTTP endpoints or Lambda functions. API Gateway sends the entire request to HTTP endpoints, including resource path, headers, query string parameters, and body. For Lambda integrations, API Gateway applies a default mapping to send all of the request information and responses follow a default interface. To learn more read our [documentation](#)

The screenshot shows the configuration for an ANY method. The 'Integration type' is set to 'HTTP Proxy'. The 'Endpoint URL' field contains 'emo-endpoint.execute-api.com/{proxy}'. The 'Content Handling' dropdown is set to 'Passthrough'. A blue 'Save' button is visible at the bottom right.

Na API recém-criada, o caminho de recurso de proxy da API de {{proxy+}} torna-se o espaço reservado de qualquer um dos endpoints do back-end em `http://petstore-demo-endpoint.execute-api.com/`. Por exemplo, ele pode ser `petstore`, `petstore/pets` e `petstore/pets/{petId}`. O método ANY serve como um espaço reservado para qualquer um dos verbos HTTP com suporte em tempo de execução.

Testar uma API com a integração de proxy HTTP

O êxito de determinada solicitação do cliente depende do seguinte:

- Se o back-end disponibilizou o endpoint de back-end 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 back-end para analisar o resultado corretamente. O API Gateway não se interpõe para facilitar as interações entre o cliente e o back-end.

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

1. Para usar o console do API Gateway para testar a invocação da API, faça o seguinte:

- a. Escolha ANY em um recurso de proxy na árvore Recursos.
- b. Escolha Test no painel Method Execution.
- c. Na lista suspensa Method, escolha um verbo HTTP com suporte do back-end.
- d. Em Caminho, digite um caminho específico para o recurso de proxy compatível com a operação escolhida.
- e. Se necessário, digite uma expressão de consulta compatível para a operação escolhida abaixo do cabeçalho Query Strings.
- f. Se necessário, digite uma ou mais expressões de cabeçalho compatíveis para a operação escolhida abaixo do cabeçalho Headers.
- g. Se configurado, defina os valores variáveis de estágio para a operação escolhida abaixo do cabeçalho Stage Variables.
- h. Se solicitado e necessário, escolha um certificado de cliente gerado pelo API Gateway abaixo do cabeçalho Client Certificate para a operação a ser autenticada pelo back-end.
- i. Se solicitado, digite um corpo de solicitação apropriado no editor de texto abaixo do cabeçalho Request Body.
- j. Escolha Test para testar a invocação do método.

Neste tutorial, use `GET` para Method (Método) no lugar de `ANY`, use `petstore/pets` para Path (Caminho) no lugar do caminho do `{proxy}` e `type=fish` para Query Strings (Strings de consulta).

[Method Execution](#) /{proxy+} - ANY - Method Test

Make a test call to your method with the provided input

Method

GET

Path

{proxy}

petstore/pets

Query Strings

{proxy}

type=fish

Headers

{proxy}

Use a colon (:) to separate header name and value, and new lines to declare multiple headers. eg.
Accept:application/json.

Stage Variables

No [stage variables](#) exist for this method.

Client Certificate

No client certificates have been generated.

Request Body

Request Body is not supported for GET methods.

Como o site de back-end 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 back-end 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  
}
```

2. Para usar um navegador para chamar um método GET em um recurso específico da API, faça o seguinte:
 - a. Caso não o tenha feito, escolha **Implantar a API** no menu suspenso **Ações** para a API que você criou. Siga as instruções para implantar a API em um estágio específico. Observe que é exibido **Invoke URL** na página Stage Editor resultante. Esse é o URL básico da API.
 - b. Para enviar uma solicitação GET em um recurso específico, acrescente o caminho do recurso, incluindo possíveis expressões de string de consulta no valor **Invoke URL** obtido na etapa anterior, copie o URL completo na barra de endereço do navegador e escolha **Enter**.

Para este tutorial, implante a API em um estágio de **test** e acrescente **petstore/pets?type=fish** ao URL de invocação da API. Isso produz uma URL de **https://4z9giyi2c1.execute-api.us-west-2.amazonaws.com/test/petstore/pets?type=fish**.

O resultado deve ser o mesmo que o retornado ao usar **TestInvoke** no console do API Gateway.

TUTORIAL: Criar uma API com integração não proxy HTTP

Neste tutorial, você pode criar uma API desde o início 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 \(p. 60\)](#)
- [Mapear parâmetros de solicitação para uma API do API Gateway \(p. 69\)](#)
- [Mapear carga da resposta \(p. 77\)](#)

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.

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Para criar uma API, escolha Criar nova API (para criar a primeira API) ou Criar API (para criar uma API subsequente). Depois, faça o seguinte:
 - a. Escolha Nova API.
 - b. Digite um nome em Nome da API.
 - c. Opcionalmente, adicione uma breve descrição em Descrição.
 - d. Escolha Criar API.

Como resultado, uma API vazia é criada. 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.

3. Para criar o recurso /pets, selecione a raiz, escolha Actions (Ações) e depois escolha Create Resource (Criar recurso).

Digite Pets no campo Resource Name (Nome do recurso), deixe o valor Resource Path (Caminho do recurso) como determinado e escolha Enable API Gateway CORS (Habilitar CORS do API Gateway) e depois clique em Create Resource (Criar recurso).

New Child Resource

Use this page to create a new child resource for your resource.

Configure as proxy resource [?](#)

Resource Name*

Resource Path* [/](#) [pets](#)

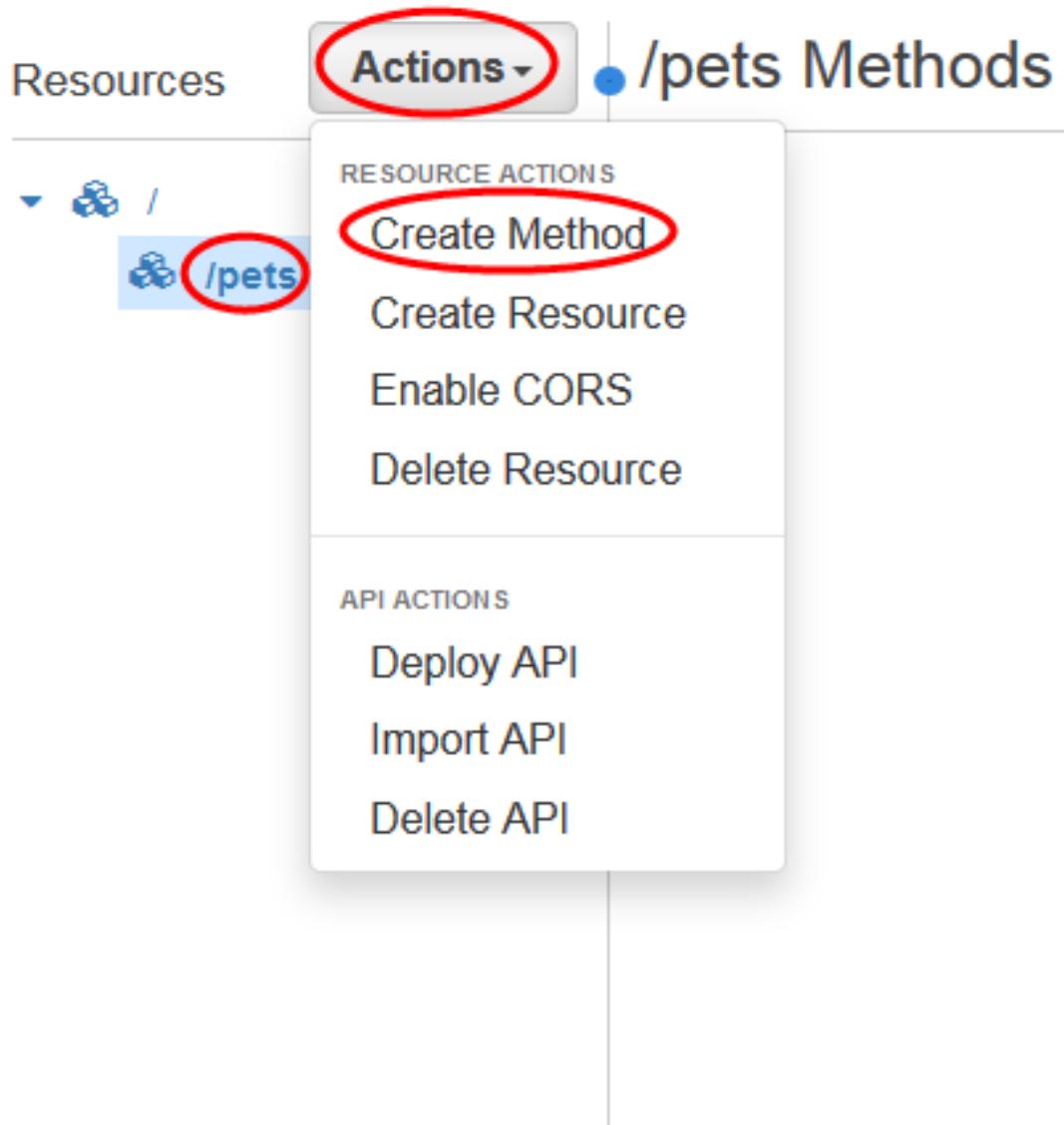
You can add path parameters using brackets. For example, the resource path `{username}` represents a path parameter called 'username'. Configuring `/{proxy+}` as a proxy resource catches all requests to its sub-resources. For example, it works for a GET request to `/foo`. To handle requests to `/`, add a new ANY method on the `/` resource.

Enable API Gateway CORS [?](#)

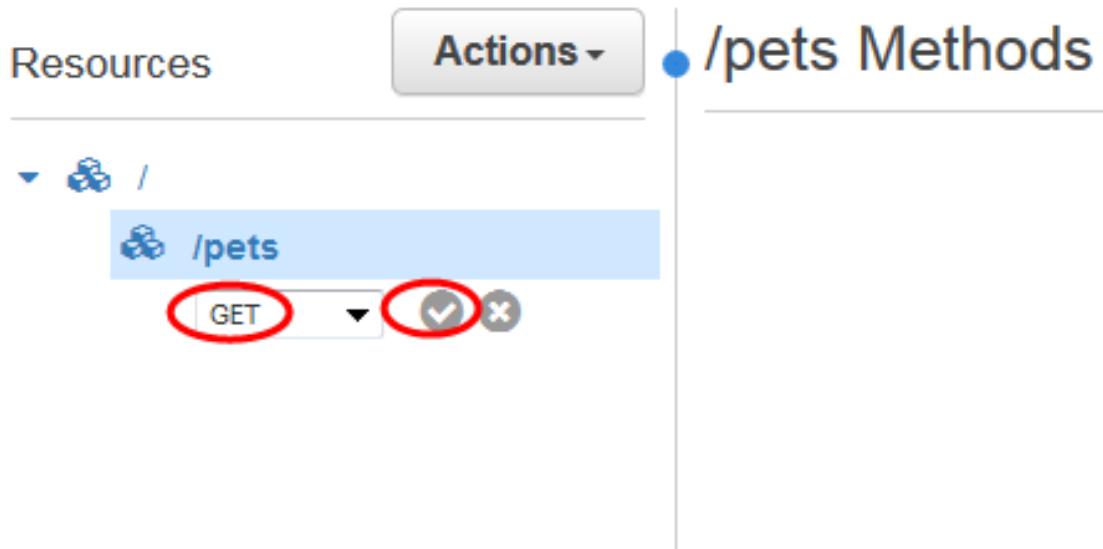
* Required

[Cancel](#) [Create Resource](#)

4. Para expor um método GET no recurso /pets, escolha Actions (Ações) e Create Method (Criar método).



Escolha GET na lista abaixo do nó de recurso /pets e escolha o ícone de marca de seleção para concluir a criação do método.



Note

Outras opções para um método de API incluem:

- 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 chamadores para obter informações sobre opções de comunicação disponíveis para o serviço de destino.

O método criado ainda não está integrado ao back-end. A próxima etapa configurará isso.

5. No painel Setup (Configuração) do método, selecione HTTP para Integration type (Tipo de integração), selecione GET na lista suspensa HTTP method (Método HTTP), digite `http://petstore-demo-endpoint.execute-api.com/petstore/pets` como o valor Endpoint URL (URL do endpoint), deixe todas as outras configurações como padrão e escolha Save (Salvar).

Note

Para o HTTP method (Método HTTP) da solicitação de integração, você deve escolher um que seja compatível com o back-end. Para `HTTP or 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.

The screenshot shows the configuration for a new method named '/pets - GET - Setup'. Under 'Integration type', 'HTTP' is selected. The 'HTTP method' is set to 'GET'. The 'Endpoint URL' is 'ndpoint.execute-api.com/petstore/pets'. The 'Content Handling' is set to 'Passthrough'. A large blue 'Save' button is visible at the bottom right.

Quando a configuração do método terminar, será exibido o painel Method Execution (Execução de método), no qual é possível configurar ainda mais a solicitação de método para adicionar uma string de consulta ou parâmetros de cabeçalho personalizados. Você também pode atualizar a solicitação de integração para mapear dados de entrada da solicitação de método para o formato exigido pelo backend.

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 determinada página. Ele usa os parâmetros de string de consulta `type` e `page` para aceitar essa entrada. Sendo assim, precisamos adicionar os parâmetros de string de consulta à solicitação de método e mapeá-los para as strings de consulta correspondentes da solicitação de integração.

6. No painel Method Execution (Execução de método) do método GET, escolha Method Request (Solicitação de método), selecione AWS_IAM para Authorization (Autorização), expanda a seção URL Query String Parameters (Parâmetros de string de consulta da URL) e escolha Add query string (Adicionar string de consulta) para criar dois parâmetros de string de consulta denominados `type` e `page`. Escolha o ícone de marca de seleção para salvar todos os parâmetros de string de consulta conforme você os adiciona.

[Method Execution](#) /pets - GET - Method Request 

Provide information about this method's authorization settings and the parameters it can receive.

Settings

Authorization AWS_IAM  

Request Validator NONE  

API Key Required false 

▼ URL Query String Parameters

Name	Required	Caching	
page	<input type="checkbox"/>	<input type="checkbox"/>	 
type	<input type="checkbox"/>	<input type="checkbox"/>	 

 [Add query string](#)

► HTTP Request Headers

► Request Body 

► SDK Settings

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 back-end. Como o método usa AWS_IAM, você deve assinar a solicitação para invocá-lo.

7. Na página Integration Request (Solicitação de integração) do método, expanda a seção URL Query String Parameters (Parâmetros de string de consulta da URL). Por padrão, os parâmetros da string de consulta da solicitação de método são mapeados para os parâmetros da string de consulta da solicitação de integração que possuem nomes semelhantes. Esse mapeamento padrão funciona para a nossa API de demonstração. Vamos deixá-los sem alterações. Para mapear um parâmetro de solicitação de método diferente para o parâmetro de solicitação de integração correspondente, escolha o ícone de lápis do parâmetro para editar a expressão de mapeamento, mostrada na coluna Mapped from (Mapeado de). Para mapear um parâmetro de solicitação de método para um parâmetro de solicitação de integração diferente, primeiro escolha o ícone de exclusão para remover o parâmetro de solicitação de integração existente e depois escolha Add query string (Adicionar string de consulta)

para especificar um novo nome e a expressão desejada para o mapeamento de parâmetro de solicitação de método.

[Method Execution](#) /pets - GET - Integration Request

Provide information about the target backend that this method will call and whether the incoming request data should be modified.

Integration type Lambda Function [i](#)
 HTTP [i](#)
 Mock [i](#)
 AWS Service [i](#)

Use HTTP Proxy integration [i](#)

HTTP method GET [e](#)

Endpoint URL <http://petstore-demo-endpoint.execute-api.com/petstore/pets> [e](#)

▶ URL Path Parameters

▼ URL Query String Parameters

Name	Mapped from i	Caching	
type	method.request.querystring.type	<input type="checkbox"/>	e
page	method.request.querystring.page	<input type="checkbox"/>	e

[+ Add query string](#)

▶ HTTP Headers

▶ Body Mapping Templates

Isso conclui a construção da API de demonstração simples. É hora de testar essa API.

8. Para testar a API usando o console do API Gateway, escolha Test (Testar) no painel Method Execution (Execução de método) para o método GET /pets. No painel Method Test (Teste de método), digite **Dog** e **2** para as strings de consulta **type** (tipo) e **page** (página), respectivamente, e escolha Test (Testar).

[Method Execution](#) /pets - GET - **Method Test**

Make a test call to your method with the provided input

Path

No path parameters exist for this resource. You can define path parameters by using the syntax `{myPathParam}` in a resource path.

Query Strings

type

Dog

page

2

Headers

No header parameters exist for this method. You can add them via Method Request.

Stage Variables

No stage variables exist for this method.

Client Certificate

No client certificates have been generated.

Request Body

Request Body is not supported for GET methods.



O resultado é mostrado da seguinte forma. (Talvez você precise rolar para baixo para ver o resultado do teste.)

Request: /pets?type=Dog&page=2

Status: 200

Latency: 1036 ms

Response Body

```
[  
  {  
    "id": 4,  
    "type": "Dog",  
    "price": 999.99  
  },  
  {  
    "id": 5,  
    "type": "Dog",  
    "price": 249.99  
  },  
  {  
    "id": 6,  
    "type": "Dog",  
    "price": 49.97  
  }  
]
```

Response Headers

```
{"Content-Type": "application/json"}
```

Logs

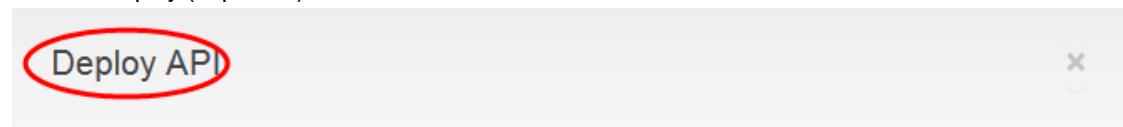
```
Execution log for request test-request  
Mon Apr 04 05:48:01 UTC 2016 : Starting execution for request: test-invoke-request  
Mon Apr 04 05:48:01 UTC 2016 : HTTP Method: GET, Resource Path: /pets  
Mon Apr 04 05:48:01 UTC 2016 : Method request path: {}  
Mon Apr 04 05:48:01 UTC 2016 : Method request query string: {page=2, type=Dog}  
Mon Apr 04 05:48:01 UTC 2016 : Method request headers: {}  
Mon Apr 04 05:48:01 UTC 2016 : Method request body before transformations: null
```

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

9. Para implantar a API, selecione-a e depois escolha Deploy API (Implantar API) no menu suspenso Actions (Ações).

The screenshot shows the AWS Lambda interface. On the left, there's a sidebar with options like APIs, Resources, Stages, Custom Authorizers, Models, PetStore, API Keys, Custom Domain Names, Client Certificates, and Settings. Under APIs, 'myApi' is selected and circled in red. In the main area, under Resources, there's a tree view showing a root resource '/' with a 'GET' method. A context menu is open over the '/' resource, with the 'Actions' dropdown expanded. The 'Deploy API' option is highlighted with a red circle.

Na caixa de diálogo Deploy API (Implantar API), escolha um estágio (ou [New Stage] para a primeira implantação da API), insira um nome (por exemplo, "test", "prod", "dev" etc.) no campo de entrada Stage name (Nome do estágio), forneça opcionalmente uma descrição em Stage description (Descrição do estágio) e/ou Deployment description (Descrição da implantação) e, em seguida, escolha Deploy (Implantar).



Choose a stage where your API will be deployed. For example, a test version of your API could be deployed to a stage named beta.

The screenshot shows the 'Deploy API' dialog box with several fields highlighted with red circles: 'Deployment stage' (dropdown), 'Stage name*' (input), 'Stage description' (input), and 'Deployment description' (input). The 'Deploy' button is also circled in red.

Uma vez implantada, você pode obter as URLs de invocação (Invoke URL (Invocar URL)) dos endpoints da API.

Se o método GET oferecesse suporte para acesso aberto (se o tipo de autorização do método estivesse definido como `NONE`), você poderia clicar duas vezes no link Invoke URL (Invocar URL) para invocar esse método em seu navegador padrão. Se necessário, você também poderia anexar parâmetros de string de consulta necessários à URL de invocação. Com o tipo de autorização `AWS_IAM` descrito aqui, você deve assinar a solicitação com um [ID de chave de acesso e a chave secreta correspondente](#) de um usuário do IAM da sua conta da AWS. Para fazer isso, você deve usar um cliente que ofereça suporte aos protocolos [Signature Version 4](#) (SigV4). Um exemplo desse cliente é um aplicativo que usa um dos SDKs da AWS ou o aplicativo [Postman](#) ou comandos cURL. Para chamar um método POST, PUT ou PATCH que aceitam uma carga, você também precisa usar esse cliente para lidar com a carga.

Para invocar esse método de API no Postman, anexe os parâmetros da sequência de consulta à URL de invocação de método específica ao estágio (conforme mostra a imagem anterior) para criar a URL completa de solicitação de método:

```
https://api-id.execute-api.region.amazonaws.com/test/pets?type=Dog&page=2
```

Especifique essa URL na barra de endereços do navegador. Escolha GET como o verbo HTTP. Selecione AWS Signature (Assinatura da AWS) para a opção Type (Tipo) na guia Authorization (Autorização) e especifique as seguintes propriedades necessárias antes de enviar a solicitação:

- Para AccessKey, digite a chave de acesso da AWS do chamador, provisionada pelo [IAM da AWS](#).
- Para SecretKey, digite a chave secreta da AWS do chamador, provisionada pelo IAM da AWS quando a chave de acesso foi inicialmente criada.
- Para AWS Region (Região da AWS), digite a Região da AWS de hospedagem de APIs, especificada na URL de invocação.
- Para Service Name (Nome do serviço), digite `execute-api` para o serviço de execução do API Gateway.

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.

Mapear parâmetros de solicitação para uma API do API Gateway

Nesta demonstração, descrevemos como mapear parâmetros de solicitação de método para os parâmetros de solicitação de integração correspondentes de uma API do API Gateway. Criamos um exemplo de API com integração HTTP personalizada e o utilizamos para demonstrar como usar o API Gateway para mapear um parâmetro de solicitação de método ao parâmetro de solicitação de integração correspondente. Em seguida, acessamos o seguinte endpoint HTTP de acesso público:

```
http://petstore-demo-endpoint.execute-api.com/petstore/pets
```

Se você copiar o URL acima, colá-lo na barra de endereços de um navegador da web e pressionar `Enter` ou `Return`, obterá o seguinte corpo de resposta formatado em JSON:

```
[
```

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

O endpoint anterior pode usar dois parâmetros de consulta: `type` e `page`. Por exemplo, altere o URL para o seguinte:

```
http://petstore-demo-endpoint.execute-api.com/petstore/pets?type=cat&page=2
```

Você receberá a seguinte carga de resposta em formato JSON, exibindo a página 2 apenas dos gatos:

```
[  
  {  
    "id": 4,  
    "type": "cat",  
    "price": 999.99  
  },  
  {  
    "id": 5,  
    "type": "cat",  
    "price": 249.99  
  },  
  {  
    "id": 6,  
    "type": "cat",  
    "price": 49.97  
  }  
]
```

Esse endpoint também oferece suporte ao uso de um ID de item, expresso por um parâmetro de caminho de URL. Por exemplo, navegue até o seguinte:

```
http://petstore-demo-endpoint.execute-api.com/petstore/pets/1
```

As seguintes informações em formato JSON sobre o item com um ID de 1 são exibidas:

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

Além de oferecer suporte a operações GET, esse endpoint usa solicitações POST com uma carga. Por exemplo, use [Postman](#) para enviar uma solicitação de método POST para o seguinte:

```
http://petstore-demo-endpoint.execute-api.com/petstore/pets
```

incluir o cabeçalho `Content-type: application/json` e o seguinte corpo de solicitação:

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

Você receberá o seguinte objeto JSON no corpo da resposta:

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

Agora, expomos estes e outros recursos com a criação de uma API do API Gateway com a integração HTTP personalizada desse site PetStore. As tarefas incluem o seguinte:

- Criar uma API com um recurso de `https://my-api-id.execute-api.region-id.amazonaws.com/test/petstorewalkthrough/pets` atuando como um proxy para o endpoint HTTP de `http://petstore-demo-endpoint.execute-api.com/petstore/pets`.
- Permitir que a API aceite dois parâmetros de consulta de solicitação de método de `petType` e `petsPage`, mapeá-los para os parâmetros de consulta `type` e `page` da solicitação de integração, respectivamente, e transmitir a solicitação ao endpoint HTTP.
- Oferecer suporte a um parâmetro de caminho de `{petId}` na URL da solicitação de método da API para especificar um ID de item, mapeá-lo para o parâmetro de caminho `{id}` na URL da solicitação de integração e enviar a solicitação ao endpoint HTTP.
- Permitir que a solicitação de método aceite a carga JSON do formato definido pelo site de back-end e passe a carga sem modificações por meio da solicitação de integração para o endpoint HTTP de back-end.

Tópicos

- [Pré-requisitos \(p. 71\)](#)
- [Etapa 1: criar recursos \(p. 72\)](#)
- [Etapa 2: Criar e testar métodos \(p. 72\)](#)
- [Etapa 3: implantar a API \(p. 75\)](#)
- [Etapa 4: testar a API \(p. 75\)](#)

Note

Preste atenção à formatação de maiúsculas e minúsculas usada nas etapas deste passo-a-passo. Digitar uma letra minúscula em vez de uma maiúscula (ou vice-versa) pode causar erros mais adiante no processo.

Pré-requisitos

Antes de iniciar este passo-a-passo, você deve fazer o seguinte:

1. Conclua as etapas em [Pré-requisitos: prepare-se para compilar uma API no API Gateway \(p. 11\)](#), incluindo a atribuição da permissão de acesso ao API Gateway para o usuário do IAM.
2. Siga pelo menos as etapas em [TUTORIAL: Criar uma API com integração não proxy HTTP \(p. 59\)](#) para criar uma nova API denominada `MyDemoAPI` no console do API Gateway.

Etapa 1: criar recursos

Nesta etapa, você cria três recursos que permitem que a API interaja com o endpoint HTTP.

Para criar o primeiro recurso

1. No painel Resources (Recursos), selecione a raiz do recurso, representada por uma única barra invertida (/), e depois escolha Create Resource (Criar recurso) no menu suspenso Actions (Ações).
2. Em Resource Name (Nome do recurso), digite **petstorewalkthrough**.
3. Em Resource Path (Caminho do recurso), aceite o padrão /petstorewalkthrough e, em seguida, escolha Create Resource (Criar recurso).

Para criar o segundo recurso

1. No painel Resources (Recursos), selecione /petstorewalkthrough e, em seguida, escolha Create Resource (Criar recurso).
2. Em Resource Name (Nome do recurso), digite **pets**.
3. Em Resource Path (Caminho do recurso), aceite o padrão /petstorewalkthrough/pets e, em seguida, escolha Create Resource (Criar recurso).

Para criar o terceiro recurso

1. No painel Resources (Recursos), selecione /petstorewalkthrough/pets e, em seguida, escolha Create Resource (Criar recurso).
2. Em Resource Name (Nome do recurso), digite **petId**. Isso é mapeado para o ID do item no endpoint HTTP.
3. Em Resource Path (Caminho do recurso), substitua petid por **{petId}**. Use chaves ({}) em torno de petId para que /petstorewalkthrough/pets/{petId} seja exibido e, em seguida, escolha Create Resource (Criar recurso).

Isso é mapeado para /petstore/pets/**my-item-id** no endpoint HTTP.

Etapa 2: Criar e testar métodos

Nesta etapa, você integra os métodos aos endpoints HTTP de back-end, mapeia os parâmetros de solicitação do método GET para os parâmetros de solicitação de integração correspondentes e testa os métodos.

Para configurar e testar o primeiro método GET

Este procedimento demonstra o seguinte:

- Crie e integre a solicitação de método de GET /petstorewalkthrough/pets com a solicitação de integração de GET `http://petstore-demo-endpoint.execute-api.com/petstore/pets`.
- Mapeie os parâmetros de consulta da solicitação de método de `petType` e `petsPage` para os parâmetros de string de consulta da solicitação de integração de `type` e `page`, respectivamente.

1. No painel Resources (Recursos), escolha /petstorewalkthrough/pets, selecione Create Method (Criar método) no menu Actions (Ações) e, em seguida, escolha GET em /pets na lista suspensa de nomes de métodos.
2. No painel /petstorewalkthrough/pets - GET - Setup, escolha HTTP para Integration type (Tipo de integração) e escolha GET para HTTP method (Método HTTP).

3. Em Endpoint URL (URL do endpoint), digite **http://petstore-demo-endpoint.execute-api.com/petstore/pets**.
4. Escolha Salvar.
5. No painel Method Execution (Execução de método), escolha Method Request (Solicitação de método) e escolha a seta ao lado de URL Query String Parameters (Parâmetros de string de consulta de URL).
6. Escolha Add query string (Adicionar string de consulta).
7. Para Nome, digite **petType**.

Isso especifica o parâmetro de consulta **petType** na solicitação de método da API.

8. Escolha o ícone de marca de seleção para terminar de criar o parâmetro de string de consulta da URL da solicitação de método.
9. Escolha Add query string (Adicionar string de consulta) novamente.
10. Para Nome, digite **petsPage**.

Isso especifica o parâmetro de consulta **petsPage** na solicitação de método da API.

11. Escolha o ícone de marca de seleção para terminar de criar o parâmetro de string de consulta da URL da solicitação de método.
12. Escolha Method Execution (Execução de método), escolha Integration Request (Solicitação de integração) e escolha a seta ao lado de URL Query String Parameters (Parâmetros de string de consulta de URL).
13. Exclua a entrada **petType** mapeada de **method.request.querystring.petType** e a entrada **petsPage** mapeada de **method.request.querystring.petsPage**. Realize esta etapa pois o endpoint exige parâmetros de string de consulta com os nomes **type** e **page** para o URL da solicitação, em vez dos valores padrão.
14. Escolha Add query string (Adicionar string de consulta).
15. Para Nome, digite **type**. Isso cria o parâmetro de string de consulta necessário para a URL da solicitação de integração.
16. Em Mapped from (Mapeado de), digite **method.request.querystring.petType**.

Isso mapeia o parâmetro de consulta **petType** da solicitação de método para o parâmetro de consulta **type** da solicitação de integração.

17. Escolha o ícone de marca de seleção para terminar de criar o parâmetro de string de consulta da URL da solicitação de integração.
18. Escolha Add query string (Adicionar string de consulta) novamente.
19. Para Nome, digite **page**. Isso cria o parâmetro de string de consulta necessário para a URL da solicitação de integração.
20. Em Mapped from (Mapeado de), digite **method.request.querystring.petsPage**.

Isso mapeia o parâmetro de consulta **petsPage** da solicitação de método para o parâmetro de consulta **page** da solicitação de integração.

21. Escolha o ícone de marca de seleção para terminar de criar o parâmetro de string de consulta da URL da solicitação de integração.
22. Escolha Method Execution (Execução de método). Na caixa Client (Cliente), escolha TEST. Na área Query Strings (Strings de consulta), para petType, digite **cat**. Para petsPage, digite **2**.
23. Escolha Test. Se bem-sucedido, Response Body (Corpo da resposta) exibe o seguinte:

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

```
        "id": 5,
        "type": "cat",
        "price": 249.99
    },
    {
        "id": 6,
        "type": "cat",
        "price": 49.97
    }
]
```

Para configurar e testar o segundo método GET

Este procedimento demonstra o seguinte:

- Crie e integre a solicitação de método de GET `/petstorewalkthrough/pets/{petId}` com a solicitação de integração de GET `http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}`.
 - Mapeie os parâmetros de caminho da solicitação de método de `petId` para os parâmetros de caminho da solicitação de integração de `id`.
1. Na lista Resources (Recursos), escolha `/petstorewalkthrough/pets/{petId}`, selecione Create Method (Criar método) no menu suspenso Actions (Ações) e, em seguida, escolha GET como o verbo HTTP para o método.
 2. No painel Setup (Configuração), escolha HTTP para Integration type (Tipo de integração) e escolha GET para HTTP method (Método HTTP).
 3. Em Endpoint URL (URL do endpoint), digite `http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}`.
 4. Escolha Salvar.
 5. No painel Method Execution (Execução de método), escolha Integration Request (Solicitação de integração) e escolha a seta próxima a URL Path Parameters (Parâmetros de caminho de URL).
 6. Escolha Add path (Adicionar caminho).
 7. Para Nome, digite `id`.
 8. Em Mapped from (Mapeado de), digite `method.request.path.petId`.

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

9. Escolha o ícone de marca de seleção para terminar de criar o parâmetro de caminho da URL.
10. Escolha Method Execution (Execução de método) e, na caixa Client (Cliente), escolha TEST. Na área Path (Caminho), em `petId`, digite `1`.
11. Escolha Test. Se bem-sucedido, Response Body (Corpo da resposta) exibe o seguinte:

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

Para configurar e testar o método POST

Este procedimento demonstra o seguinte:

- Crie e integre a solicitação de método de POST `/petstorewalkthrough/pets` com a solicitação de integração de POST `http://petstore-demo-endpoint.execute-api.com/petstore/pets`.

- Transmite a carga JSON da solicitação de método para a carga da solicitação de integração sem modificação.
1. No painel Resources (Recursos), escolha /petstorewalkthrough/pets, selecione Create Method (Criar método) no menu suspenso Actions (Ações) e, em seguida, escolha POST como o verbo HTTP para o método.
 2. No painel Setup (Configuração), escolha HTTP para Integration type (Tipo de integração) e escolha POST para HTTP method (Método HTTP).
 3. Em Endpoint URL (URL do endpoint), digite **http://petstore-demo-endpoint.execute-api.com/petstore/pets**.
 4. Escolha Salvar.
 5. No painel Method Execution (Execução de método), na caixa Client (Cliente), escolha TEST. Expanda Request Body (Corpo da solicitação) e digite o seguinte:

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

6. Escolha Test. Se bem-sucedido, Response Body (Corpo da resposta) exibe o seguinte:

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

Etapa 3: implantar a API

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

Para implantar a API

1. No painel Resources (Recursos), escolha Deploy API (Implantar API).
2. Em Deployment stage (Estágio de implantação), selecione **test**.

Note

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

3. For Deployment description (Descrição da implantação), digite **Calling HTTP endpoint walkthrough**.
4. Escolha Deploy (Implantar).

Etapa 4: testar a API

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

1. No painel Stage Editor (Editor de estágio), ao lado de Invoke URL (Invocar URL), copie a URL na área de transferência. A aparência será semelhante à seguinte:

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

2. Cole esta URL na caixa de endereço de uma nova guia do navegador.

3. Acrescente /petstorewalkthrough/pets para obter a seguinte aparência:

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

Navegue até a URL. As seguintes informações devem ser exibidas:

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

4. Depois de petstorewalkthrough/pets, digite ?**petType=cat&petsPage=2** para que fique semelhante a isso:

```
https://my-api-id.execute-api.region-id.amazonaws.com/test/petstorewalkthrough/pets?  
petType=cat&petsPage=2
```

5. Navegue até a URL. As seguintes informações devem ser exibidas:

```
[  
  {  
    "id": 4,  
    "type": "cat",  
    "price": 999.99  
  },  
  {  
    "id": 5,  
    "type": "cat",  
    "price": 249.99  
  },  
  {  
    "id": 6,  
    "type": "cat",  
    "price": 49.97  
  }  
]
```

6. Depois de petstorewalkthrough/pets, substitua ?**petType=cat&petsPage=2** por /1 para que fique semelhante a isso:

```
https://my-api-id.execute-api.region-id.amazonaws.com/test/petstorewalkthrough/pets/1
```

7. Navegue até a URL. As seguintes informações devem ser exibidas:

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

```
}
```

8. Usando uma ferramenta de proxy de depuração da Web ou a ferramenta de linha de comando cURL, envie uma solicitação de método POST à URL do procedimento anterior. Acrescente /petstorewalkthrough/pets para obter a seguinte aparência:

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

Anexe o seguinte cabeçalho:

```
Content-Type: application/json
```

Adicione o seguinte código ao corpo da solicitação:

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

Por exemplo, se você usar a ferramenta de linha de comando cURL, execute um comando semelhante ao seguinte:

```
curl -H "Content-Type: application/json" -X POST -d "{\"type\": \"dog\",
\"price\": 249.99}" https://my-api-id.execute-api.region-id.amazonaws.com/test/
petstorewalkthrough/pets
```

As seguintes informações devem ser retornadas no corpo da resposta:

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

Você chegou ao final deste passo a passo.

Mapear carga da resposta

Nesta demonstração, mostraremos como usar modelos e modelos de mapeamento no API Gateway para transformar a saída de uma chamada de API de um esquema de dados para outro. Este passo-a-passo baseia-se nas instruções e nos conceitos em [Conceitos básicos sobre o Amazon API Gateway \(p. 11\)](#) e [Mapear parâmetros de solicitação para uma API do API Gateway \(p. 69\)](#). Se você ainda não completou esses tutoriais passo-a-passo, sugerimos que faça isso primeiro.

Esta demonstração usa o API Gateway para obter dados de exemplo de um endpoint HTTP publicamente acessível e de uma função do AWS Lambda criada. Tanto o endpoint HTTP quanto a função Lambda retornam os mesmos dados do exemplo:

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

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

Você usará modelos e modelos de mapeamento para transformar esses dados em um ou mais formatos de saída. No API Gateway, um módulo define o formato, também conhecido como o schema ou forma, de alguns dados. Em API Gateway, um modelo do mapeamento é usado para transformar alguns dados de um formato a outros. Para obter mais informações, consulte [Criar modelos e modelos de mapeamento para mapeamentos de solicitação e resposta \(p. 273\)](#).

O primeiro modelo e o modelo de mapeamento são usados para renomear `id` como `number`, `type` como `class` e `price` como `salesPrice`, da seguinte maneira:

```
[  
  {  
    "number": 1,  
    "class": "dog",  
    "salesPrice": 249.99  
  },  
  {  
    "number": 2,  
    "class": "cat",  
    "salesPrice": 124.99  
  },  
  {  
    "number": 3,  
    "class": "fish",  
    "salesPrice": 0.99  
  }  
]
```

O segundo modelo e modelo de mapeamento são usados para combinar `id` e `type` em `description` e para renomear `price` como `askingPrice`, da seguinte maneira:

```
[  
  {  
    "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 terceiro modelo e modelo de mapeamento são usados para combinar `id`, `type` e `price` em um conjunto de `listings`, da seguinte maneira:

```
{
```

```
    "listings": [
        "Item 1 is a dog. The asking price is 249.99.",
        "Item 2 is a cat. The asking price is 124.99.",
        "Item 3 is a fish. The asking price is 0.99."
    ]
}
```

Tópicos

- [Etapa 1: criar modelos \(p. 79\)](#)
- [Etapa 2: criar recursos \(p. 81\)](#)
- [Etapa 3: criar métodos GET \(p. 82\)](#)
- [Etapa 4: Criar uma função do Lambda \(p. 82\)](#)
- [Etapa 5: configurar e testar os métodos \(p. 84\)](#)
- [Etapa 6: implantar a API \(p. 87\)](#)
- [Etapa 7: testar a API \(p. 87\)](#)
- [Etapa 8: Limpeza \(p. 89\)](#)

Etapa 1: criar modelos

Nesta etapa, você cria quatro modelos. Os três primeiros modelos representam os formatos de saída de dados para uso com o endpoint HTTP e a função Lambda. O último modelo representa o esquema de entrada de dados para uso com a função Lambda.

Para criar o primeiro modelo de saída

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Se MyDemoAPI for exibido, escolha Models (Modelos).
3. Escolha Criar.
4. Em Model name (Nome do modelo), digite **PetsModelNoFlatten**.
5. Em Content type (Tipo de conteúdo), digite **application/json**.
6. Em Model description (Descrição do modelo), digite **Changes id to number, type to class, and price to salesPrice**.
7. Em Model schema (Esquema do modelo), digite a seguinte definição compatível com o Esquema JSON:

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "PetsModelNoFlatten",
    "type": "array",
    "items": {
        "type": "object",
        "properties": {
            "number": { "type": "integer" },
            "class": { "type": "string" },
            "salesPrice": { "type": "number" }
        }
    }
}
```

8. Escolha Create model (Criar modelo).

Para criar o segundo modelo de saída

1. Escolha Criar.

2. Em Model name (Nome do modelo), digite **PetsModelFlattenSome**.
3. Em Content type (Tipo de conteúdo), digite **application/json**.
4. Em Model description (Descrição do modelo), digite **Combines id and type into description, and changes price to askingPrice**.
5. Em Model schema (Esquema do modelo), digite o seguinte:

```
{  
    "$schema": "http://json-schema.org/draft-04/schema#",  
    "title": "PetsModelFlattenSome",  
    "type": "array",  
    "items": {  
        "type": "object",  
        "properties": {  
            "description": { "type": "string" },  
            "askingPrice": { "type": "number" }  
        }  
    }  
}
```

6. Escolha Create model (Criar modelo).

Para criar o terceiro modelo de saída

1. Escolha Criar.
2. Em Model name (Nome do modelo), digite **PetsModelFlattenAll**.
3. Em Content type (Tipo de conteúdo), digite **application/json**.
4. Em Model description (Descrição do modelo), digite **Combines id, type, and price into a set of listings**.
5. Em Model schema (Esquema do modelo), digite o seguinte:

```
{  
    "$schema": "http://json-schema.org/draft-04/schema#",  
    "title": "PetsModelFlattenAll",  
    "type": "object",  
    "properties": {  
        "listings": {  
            "type": "array",  
            "items": {  
                "type": "string"  
            }  
        }  
    }  
}
```

6. Escolha Create model (Criar modelo).

Para criar o modelo de entrada

1. Escolha Criar.
2. Em Model name (Nome do modelo), digite **PetsLambdaModel**.
3. Em Content type (Tipo de conteúdo), digite **application/json**.
4. Em Model description (Descrição do modelo), digite **GetPetsInfo model**.
5. Em Model schema (Esquema do modelo), digite o seguinte:

```
{
```

```
 "$schema": "http://json-schema.org/draft-04/schema#",
"title": "PetsLambdaModel",
"type": "array",
"items": {
  "type": "object",
  "properties": {
    "id": { "type": "integer" },
    "type": { "type": "string" },
    "price": { "type": "number" }
  }
}
```

6. Escolha Create model (Criar modelo).

Etapa 2: criar recursos

Nesta etapa, você cria quatro recursos. Os três primeiros recursos permitem que você obtenha os dados de exemplo do endpoint HTTP nos três formatos de saída. O último recurso permite que você obtenha os dados de exemplo da função do Lambda no esquema de saída que combina `id` e `type` em `description` e renomeia `price` como `askingPrice`.

Para criar o primeiro recurso

1. Na lista de links, escolha Resources (Recursos).
2. No painel Resources (Recursos), selecione /petstorewalkthrough e, em seguida, escolha Create Resource (Criar recurso).
3. Em Resource Name (Nome do recurso), digite **NoFlatten**.
4. Em Resource Path (Caminho do recurso), aceite o padrão /petstorewalkthrough/noflatten e, em seguida, escolha Create Resource (Criar recurso).

Para criar o segundo recurso

1. No painel Resources (Recursos), selecione /petstorewalkthrough novamente e, em seguida, escolha Create Resource (Criar recurso).
2. Em Resource Name (Nome do recurso), digite **FlattenSome**.
3. Em Resource Path (Caminho do recurso), aceite o padrão /petstorewalkthrough/flattensome e, em seguida, escolha Create Resource (Criar recurso).

Para criar o terceiro recurso

1. No painel Resources (Recursos), selecione /petstorewalkthrough novamente e, em seguida, escolha Create Resource (Criar recurso).
2. Em Resource Name (Nome do recurso), digite **FlattenAll**.
3. Em Resource Path (Caminho do recurso), aceite o padrão /petstorewalkthrough/flattenall e, em seguida, escolha Create Resource (Criar recurso).

Para criar o quarto recurso

1. No painel Resources (Recursos), selecione /petstorewalkthrough novamente e, em seguida, escolha Create Resource (Criar recurso).
2. Em Resource Name (Nome do recurso), digite **LambdaFlattenSome**.
3. Em Resource Path (Caminho do recurso), aceite o padrão /petstorewalkthrough/lambdaflattensome e, em seguida, escolha Create Resource (Criar recurso).

Etapa 3: criar métodos GET

Nesta etapa, você cria um método GET para cada um dos recursos que criou na etapa anterior.

Para criar o primeiro método GET

1. Na lista Resources (Recursos), selecione /petstorewalkthrough/flattenall e, em seguida, escolha Create Method (Criar método).
2. Na lista suspensa, escolha GET e, em seguida, escolha o ícone de marca de seleção para salvar sua opção.
3. No painel de configuração, selecione HTTP para o Integration type (Tipo de integração) e GET para HTTP method (Método HTTP), digite **http://petstore-demo-endpoint.execute-api.com/petstore/pets** em Endpoint URL (URL do endpoint) e escolha Save (Salvar).

Para criar o segundo método GET

1. Na lista Resources (Recursos), selecione /petstorewalkthrough/lambdaflattensome e, em seguida, escolha Create Method (Criar método).
2. Na lista suspensa, escolha GET e, em seguida, escolha a marca de seleção para salvar sua opção.
3. No painel de configuração, escolha Lambda Function (Função do Lambda) para o Integration type (Tipo de integração), escolha a região em que a [função GetPetsInfo do Lambda \(p. 82\)](#) foi criada na lista suspensa Lambda Region (Região do Lambda), escolha **GetPetsInfo** em Lambda Function (Função do Lambda) e selecione Save (Salvar). Escolha OK quando solicitado a adicionar permissão para a função do Lambda.

Para criar o terceiro método GET

1. Na lista Resources (Recursos), selecione /petstorewalkthrough/flattensome e, em seguida, escolha Create Method (Criar método).
2. Na lista suspensa, escolha GET e, em seguida, escolha o ícone de marca de seleção para salvar sua opção.
3. No painel de configuração, selecione HTTP para o Integration type (Tipo de integração) e GET para HTTP method (Método HTTP), digite **http://petstore-demo-endpoint.execute-api.com/petstore/pets** em Endpoint URL (URL do endpoint) e, em seguida, escolha Save (Salvar).

Para criar o quarto método GET

1. Na lista Resources (Recursos), selecione /petstorewalkthrough/noflatten e, em seguida, escolha Actions (Ações), Create Method (Criar método).
2. Na lista suspensa, escolha GET e, em seguida, escolha o ícone de marca de seleção para salvar sua opção.
3. No painel de configuração, selecione HTTP para o Integration type (Tipo de integração) e GET para HTTP method (Método HTTP), digite **http://petstore-demo-endpoint.execute-api.com/petstore/pets** em Endpoint URL (URL do endpoint) e, em seguida, escolha Save (Salvar).

Etapa 4: Criar uma função do Lambda

Nesta etapa, você cria uma função Lambda que retorna os dados da amostra.

Para criar a função do Lambda

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

2. Faça uma das coisas a seguir:
 - Se uma página de boas-vindas for exibida, escolha Get Started Now (Comece a usar agora).
 - Se a página Lambda: Function list (Lambda: lista de funções) for exibida, escolha Create a Lambda function (Criar uma função do Lambda).
3. Para Nome, digite **GetPetsInfo**.
4. Em Description (Descrição), digite **Gets information about pets**.
5. Em Code template (Modelo de código), escolha None (Nenhum).
6. Digite o seguinte código:

```
console.log('Loading event');

exports.handler = function(event, context, callback) {
  callback(null,
    [{"id": 1, "type": "dog", "price": 249.99},
     {"id": 2, "type": "cat", "price": 124.99},
     {"id": 3, "type": "fish", "price": 0.99}]); // SUCCESS with message
};
```

Tip

No código anterior, escrito em Node.js, o `console.log` grava informações em um log do Amazon CloudWatch. O `event` contém a entrada da função Lambda. O `context` contém o contexto de chamada. O `callback` retorna o resultado. Para obter mais informações sobre como escrever código de funções do Lambda, consulte a seção "Modelo de programação" em [AWS Lambda: como funciona](#) e as amostras passo a passo no Guia do desenvolvedor do AWS Lambda.

7. Em Handler name (Nome do handler), deixe o padrão de `index.handler`.
8. Em Role (Função), escolha a função de execução do Lambda, `APIGatewayLambdaExecRole`, criada em [Criar uma API do API Gateway com integração à Lambda \(p. 28\)](#).
9. Escolha Create Lambda function (Criar função do Lambda).
10. Na lista de funções, escolha `GetPetsInfo` para mostrar os detalhes da função.
11. Anote a região da AWS na qual você criou essa função. Você precisará disso mais tarde.
12. Na lista pop-up, escolha Edit or test function (Editar ou testar função).
13. Em Sample event (Evento de exemplo), substitua qualquer código exibido pelo seguinte:

```
{  
}
```

Tip

As chaves vazias significam que não há valores de entrada para essa função Lambda. Essa função simplesmente retorna o objeto JSON que contém as informações sobre animais de estimativa, portanto, esses pares de valores-chave não são necessários aqui.

14. Escolha Invoke (Invocar). Execution result (Resultado da execução) mostra `[{"id":1,"type":"dog","price":249.99}, {"id":2,"type":"cat","price":124.99}, {"id":3,"type":"fish","price":0.99}]`, que também é gravado nos arquivos de log do CloudWatch Logs.
15. Escolha Go to function list (Ir para a lista de funções).

Etapa 5: configurar e testar os métodos

Nessa etapa, você configura as respostas de método, as solicitações de integração e as respostas de integração para especificar os esquemas (ou modelos) de dados de entrada e saída para os métodos GET associados ao endpoint HTTP e à função Lambda. Você também aprende a testar a chamada desses métodos usando o console do API Gateway.

Para configurar a integração para o primeiro método GET e testá-lo em seguida

1. Na árvore Resources (Recursos) da API, escolha GET no nó /petstorewalkthrough/flattenall.
2. No painel Method Execution (Execução de método), escolha Method Response (Resposta de método) e escolha a seta ao lado de 200.
3. Na área Response Models for 200 (Modelos de resposta para 200), para application/json, escolha o ícone de lápis para começar a configurar o modelo para a saída do método. Em Models (Modelos), escolha PetsModelFlattenAll e selecione o ícone de marca de verificação para salvar a configuração.
4. Escolha Method Execution (Execução de método), Integration Response (Resposta de integração) e escolha a seta ao lado de 200.
5. Expanda a seção Body Mapping Templates (Modelos de mapeamento do corpo) e escolha application/json em Content-Type (Tipo de conteúdo).
6. Em Generate template from model (Gerar modelo a partir do modelo), escolha PetsModelFlattenAll para exibir um modelo de mapeamento após o modelo PetsModelFlattenAll como um ponto de partida.
7. Modifique o código do modelo de mapeamento da seguinte maneira:

```
#set($inputRoot = $input.path('$'))  
{  
    "listings" : [  
#foreach($elem in $inputRoot)  
        "Item number $elem.id is a $elem.type. The asking price is  
        $elem.price."#if($foreach.hasNext),#end  
  
#end  
    ]  
}
```

8. Escolha Salvar.
9. Escolha Method Execution (Execução de método) e na caixa Client (Cliente), escolha TEST e selecione Test (Testar). Se bem-sucedido, Response Body (Corpo da solicitação) exibe o seguinte:

```
{  
    "listings" : [  
        "Item number 1 is a dog. The asking price is 249.99.",  
        "Item number 2 is a cat. The asking price is 124.99.",  
        "Item number 3 is a fish. The asking price is 0.99."  
    ]  
}
```

Para configurar a integração para o segundo método GET e testá-lo em seguida

1. Na árvore Resources (Recursos) da API, escolha GET no nó /petstorewalkthrough/lambdaflattensome.
2. Em Method Execution (Execução de método), selecione Method Response (Resposta de método). Em seguida, escolha a seta ao lado de 200 para expandir a seção.
3. Na área Response Models for 200 (Modelos de resposta para 200), escolha o ícone de lápis na linha para o tipo de conteúdo de aplicativo/json. Escolha PetsModelFlattenSome em Models (Modelos) e, em seguida, selecione o ícone de marca de verificação para salvar a opção.

4. Volte para Method Execution (Execução de método). Escolha Integration Response (Resposta de integração) e depois selecione a seta ao lado de 200.
5. Na seção Body Mapping Templates (Modelos de mapeamento do corpo), escolha application/json em Content-Type (Tipo de conteúdo).
6. Em Generate template (Gerar modelo), escolha PetsModelFlattenSome para exibir o modelo de script de mapeamento para a saída desse método.
7. Modifique o código da seguinte maneira e escolha Save (Salvar):

```
#set($inputRoot = $input.path('$'))  
[  
#foreach($elem in $inputRoot)  
{  
    "description" : "Item $elem.id is a $elem.type.",  
    "askingPrice" : $elem.price  
}#if($foreach.hasNext),#end  
  
#end  
]
```

8. Escolha Method Execution (Execução de método) e na caixa Client (Cliente), escolha TEST e selecione Test (Testar). Se bem-sucedido, Response Body (Corpo da solicitação) exibe o seguinte:

```
[  
{  
    "description" : "Item 1 is a dog.",  
    "askingPrice" : 249.99  
,  
{  
    "description" : "Item 2 is a cat.",  
    "askingPrice" : 124.99  
,  
{  
    "description" : "Item 3 is a fish.",  
    "askingPrice" : 0.99  
}  
]
```

Para configurar a integração para o terceiro método GET e testá-lo em seguida

1. Na árvore Resources (Recursos) da API, escolha GET no nó /petstorewalkthrough/flattensome.
2. No painel Method Execution (Execução de), selecione Method Response (Resposta de método).
3. Escolha a seta ao lado de 200.
4. Na área Response Models for 200 (Modelos de resposta para 200), para application/json, escolha o ícone de lápis. Em Models (Modelos), escolha PetsModelFlattenSome e, em seguida, selecione o ícone de marca de verificação para salvar a opção.
5. Volte para Method Execution (Execução de método) e escolha Integration Response (Resposta de integração).
6. Escolha a seta ao lado de 200 para expandir a seção.
7. Expanda a área Body Mapping Templates (Modelos de mapeamento do corpo). Escolha application/json para Content-Type (Tipo de conteúdo). Em Generate template (Gerar modelo), escolha PetsModelFlattenSome para exibir um modelo de script de mapeamento para a saída desse método.
8. Modifique o código da seguinte maneira:

```
#set($inputRoot = $input.path('$'))  
[
```

```
#foreach($elem in $inputRoot)
{
    "description": "Item $elem.id is a $elem.type.",
    "askingPrice": $elem.price
}#if($foreach.hasNext),#end

#end
]
```

9. Escolha Salvar.
10. Volte para Method Execution (Execução de método) e escolha TEST na caixa Client (Cliente). Em seguida, escolha Test (Testar). Se bem-sucedido, Response Body (Corpo da solicitação) exibe o seguinte:

```
[
{
    "description": "Item 1 is a dog.",
    "askingPrice": 249.99
},
{
    "description": "Item 2 is a cat.",
    "askingPrice": 124.99
},
{
    "description": "Item 3 is a fish.",
    "askingPrice": 0.99
}
]
```

Para configurar a integração para o quarto método GET e testá-lo em seguida

1. Na árvore Resources (Recursos) da API, escolha GET no nó /petstorewalkthrough/noflatten.
2. No painel Method Execution (Execução de método), escolha Method Response (Resposta de método) e depois expanda a seção 200.
3. Na área Response Models for 200 (Modelos de resposta para 200), para application/json, escolha o ícone de lápis para atualizar o modelo de resposta para esse método.
4. Escolha PetsModelNoFlatten como o modelo para o tipo de conteúdo de application/json e escolha o ícone de marca de verificação para salvar a escolha.
5. Escolha Method Execution (Execução de método), Integration Response (Resposta de integração) e escolha a seta ao lado de 200 para expandir a seção.
6. Expanda a seção Mapping Templates (Modelos de mapeamento). Escolha application/json para Content-Type (Tipo de conteúdo). Em Generate templates (Gerar modelos), escolha PetsModelNoFlatten para exibir um modelo de script de mapeamento para a saída desse método.
7. Modifique o código da seguinte maneira:

```
#set($inputRoot = $input.path('$'))
[
#foreach($elem in $inputRoot)
{
    "number": $elem.id,
    "class": "$elem.type",
    "salesPrice": $elem.price
}#if($foreach.hasNext),#end

#end
]
```

8. Escolha Salvar.

- Volte para Method Execution (Execução de método) e, na caixa Client (Cliente), escolha TEST e selecione Test (Testar). Se bem-sucedido, Response Body (Corpo da solicitação) exibe o seguinte:

```
[  
 {  
   "number": 1,  
   "class": "dog",  
   "salesPrice": 249.99  
 },  
 {  
   "number": 2,  
   "class": "cat",  
   "salesPrice": 124.99  
 },  
 {  
   "number": 3,  
   "class": "fish",  
   "salesPrice": 0.99  
 }  
 ]
```

Etapa 6: implantar a API

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

Para implantar a API

- No painel Resources (Recursos), escolha Deploy API (Implantar API).
- Em Deployment stage (Estágio de implantação), selecione test.
- For Deployment description (Descrição da implantação), digite **Using models and mapping templates walkthrough**.
- Escolha Deploy (Implantar).

Etapa 7: testar a API

Nesta etapa, você sai do console do API Gateway para interagir com o endpoint HTTP e com a função do Lambda.

- No painel Stage Editor (Editor de estágio), ao lado de Invoke URL (Invocar URL), copie a URL na área de transferência. A aparência será semelhante à seguinte:

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

- Cole esta URL na caixa de endereço de uma nova guia do navegador.
- Acrescente /petstorewalkthrough/noflatten para obter a seguinte aparência:

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

Navegue até a URL. As seguintes informações devem ser exibidas:

```
[  
 {  
   "number": 1,  
   "class": "dog",  
   "salesPrice": 249.99  
 }
```

```
        },
        {
            "number": 2,
            "class": "cat",
            "salesPrice": 124.99
        },
        {
            "number": 3,
            "class": "fish",
            "salesPrice": 0.99
        }
    ]
```

4. Depois de petstorewalkthrough/, substitua noflatten por flattensome.
5. Navegue até a URL. As seguintes informações devem ser exibidas:

```
[{
    {
        "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
    }
}]
```

6. Depois de petstorewalkthrough/, substitua flattensome por flattenall.
7. Navegue até a URL. As seguintes informações devem ser exibidas:

```
{
    "listings" : [
        "Item number 1 is a dog. The asking price is 249.99.",
        "Item number 2 is a cat. The asking price is 124.99.",
        "Item number 3 is a fish. The asking price is 0.99."
    ]
}
```

8. Depois de petstorewalkthrough/, substitua flattenall por lambdaflattensome.
9. Navegue até a URL. As seguintes informações devem ser exibidas:

```
[{
    {
        "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
    }
}]
```

Etapa 8: Limpeza

Se você não precisar mais da função Lambda que criou para este passo-a-passo, poderá excluí-la agora. Você também pode excluir os recursos do IAM acompanhantes.

Warning

Se você excluir uma função Lambda da qual as suas APIs dependem, essas APIs deixarão de funcionar. A exclusão de uma função Lambda não pode ser desfeita. Se quiser usar a função Lambda novamente, você deverá recriar essa função.

Se você excluir um recurso do IAM do qual uma função do Lambda depende, a função do Lambda deixará de funcionar, juntamente com as APIs que dependem dela. A exclusão de um recurso do IAM não pode ser desfeita. Se quiser usar o recurso IAM novamente, você deverá recriá-lo. Se você planeja continuar a testar os recursos criados para este e outros tutoriais passo-a-passo, não exclua a função de invocação do Lambda ou a função de execução do Lambda.

Para excluir a função 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 página Lambda: Function list (Lambda: lista de funções), na lista de funções, escolha o botão ao lado de GetPetsInfo e escolha Actions (Ações), Delete (Excluir). Quando solicitado, escolha Delete (Excluir) novamente.

Para 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 APIGatewayLambdaExecRole e, em seguida, escolha Role Actions (Ações da função), Delete Role (Excluir função). Quando solicitado, escolha Sim, excluir.
4. Na área Details (Detalhes), escolha Policies (Políticas).
5. Selecione APIGatewayLambdaExecPolicy e, em seguida, escolha Policy Actions (Ações da política), Delete (Excluir). Quando solicitado, escolha Delete (Excluir).

Você chegou ao final deste passo a passo.

TUTORIAL: Criar uma API com integração privada do API Gateway

Você pode criar uma API do API Gateway com integração privada para fornecer aos clientes o acesso a recursos de HTTP/HTTPS dentro da Amazon Virtual Private Cloud (Amazon VPC). Esses recursos da VPC são endpoints de HTTP/HTTPS em uma instância do EC2 por trás de um Load balancer de rede na VPC. O Load balancer de rede encapsula o recurso da VPC e roteia as solicitações recebidas ao recurso de destino.

Quando um cliente chama a API, o API Gateway conecta-se ao Load balancer de rede por meio do link da VPC pré-configurado. Um link de VPC é encapsulado por um recurso do API Gateway de [VpcLink](#). Ele é responsável por encaminhar as solicitações do método da API aos recursos da VPC e retornar respostas de back-end ao chamador. Para um desenvolvedor de API, um [VpcLink](#) é funcionalmente equivalente a um endpoint de integração.

Para criar uma API com integração privada, é necessário criar outro [VpcLink](#) ou escolher um existente que esteja conectado a um Load balancer de rede e vise os recursos da VPC desejados. Você deve ter [permissões apropriadas](#) (p. 253) para criar e gerenciar um [VpcLink](#). Em seguida, configure um [método](#)

de API e integre-o ao VpcLink, configurando HTTP ou HTTP_PROXY como o [tipo de integração](#), definindo VPC_LINK como o [tipo de conexão](#) da integração e configure o identificador do VpcLink na integração [connectionId](#).

Note

O Load balancer de rede e a API devem pertencer à mesma conta da AWS.

Para começar rapidamente a criação de uma API para acessar recursos da VPC, explicaremos as etapas essenciais para a criação de uma API com a integração privada, usando o console do API Gateway. Antes de criar a API, faça o seguinte:

1. Crie um recurso da VPC, crie ou escolha um Load balancer de rede na conta na mesma região e adicione a instância do EC2 que hospeda o recurso como destino do Load balancer de rede. Para obter mais informações, consulte [Configurar um Load balancer de rede para Integrações privadas do API Gateway \(p. 253\)](#).
2. Conceda permissões para criar os links da VPC para integrações privadas. Para obter mais informações, consulte [Conceder permissões para criar um link de VPC \(p. 253\)](#).

Após criar o recurso da VPC e o Load balancer de rede com o recurso da VPC configurado em seus grupos de destino, siga as instruções a seguir para criar uma API e integrá-la ao recurso da VPC por meio de um VpcLink em uma integração privada.

Para criar uma API com a integração privada usando o console do API Gateway

1. Faça login no console do API Gateway e escolha uma região; , por exemplo, us-west-2, na barra de navegação.
2. Crie um link da VPC, se você ainda não o fez:
 - a. No painel de navegação principal, escolha VPC Links (Links de VPC) e, em seguida, selecione + Create (+ Criar).
 - b. Em VPC Link (Link de VPC), digite um nome (por exemplo, my-test-vpc-link) no campo Name (Nome).
 - c. Opcionalmente, faça uma descrição do link da VPC na área de texto Description (Descrição).
 - d. Escolha um Load balancer de rede na lista suspensa Target NLB (NLB de destino).

É necessário ter o Load balancer de rede já criado na região escolhida para que o Load balancer de rede esteja presente na lista.

- e. Selecione Create (Criar) para começar a criar o link da VPC.

A resposta inicial retorna uma representação de recurso do VpcLink com o ID do link da VPC e um status de PENDING. Isso ocorre porque a operação é assíncrona e leva cerca de 2 a 4 minutos para ser concluída. Após a conclusão bem-sucedida, o status fica AVAILABLE. Enquanto isso, você pode continuar a criar a API.

3. Escolha APIs no painel de navegação principal e, em seguida, escolha + Create API (+ Criar API) para criar uma nova API de um tipo de endpoint otimizado para fronteiras ou regional.
4. Para o recurso raiz (/), escolha Create Method (Criar método) no menu suspenso Actions (Ações) e, em seguida, escolha GET.
5. No painel / GET - Setup, inicialize a integração do método da API da seguinte forma:
 - a. Escolha VPC Link para Integration type (Tipo de integração).
 - b. Escolha Use Proxy Integration (Usar a integração de proxy).
 - c. Na lista suspensa Method (Método), escolha GET como o método de integração.
 - d. Na lista suspensa VPC Link (Link de VPC), escolha [Use Stage Variables] e digite \${stageVariables.vpcLinkId} na caixa de texto abaixo.

Vamos definir a variável do estágio `vpcLinkId` após implantar a API em um estágio e definir seu valor para o ID do `VPC Link` criado na Step 1 (Etapa 1).

- e. Digite um URL, por exemplo, `http://myApi.example.com`, para a Endpoint URL (URL do endpoint).

Aqui, o nome do host (por exemplo, `myApi.example.com`) é usado para definir o cabeçalho de Host da solicitação de integração.

Note

Para o Load balancer de rede (NLB), use o nome de DNS do NLB conforme descrito em [Conceitos básicos de Load balancer de redes](#).

- f. Deixe a seleção Use Default Timeout (Usar o tempo limite padrão) como está, a menos que você queira personalizar os tempos limites de integração.
 - g. Escolha Save (Salvar) para concluir a configuração da integração.
- Com a integração de proxy, a API está pronta para a implantação. Caso contrário, você precisa continuar a configuração de respostas apropriadas do método e da integração.
- h. No menu suspenso Actions (Ações), selecione Deploy API (Implantar API) e, em seguida, escolha um estágio novo ou já existente para implantar a API.
- Anote a Invoke URL (Invocar URL) resultante. Ela é necessária para invocar a API. Antes de fazer isso, você deve configurar a variável de estágio do `vpcLinkId`.
- i. No painel Stage Editor (Editor de estágio), escolha a guia Stage Variables (Variáveis de estágio) e escolha Add Stage Variable (Adicionar variável de estágio).
 - i. Na coluna Name (Nome), digite `vpcLinkId`.
 - ii. Na coluna Value (Valor), digite o ID do VPC_LINK, por exemplo, `gix6s7`.
 - iii. Escolha o ícone de marca de seleção para salvar esta variável de estágio.

Usando a variável de estágio, você pode facilmente alternar para diferentes links de VPC para a API alterando o valor da variável do estágio.

Isso conclui a criação da API. Você pode testar a invocação da API com outras integrações.

TUTORIAL: Criar uma API do API Gateway com integração da AWS

Os tópicos [TUTORIAL: Criar uma API Hello World com integração de proxy do Lambda \(p. 29\)](#) e [Criar uma API do API Gateway com integração à Lambda \(p. 28\)](#) descrevem como criar uma API do API Gateway para expor a função do Lambda integrada. Além disso, você pode criar uma API do API Gateway para expor outros serviços da AWS, tais como o 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 são casos especiais, onde a chamada de função do Lambda é exposta através 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 aplicativo 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 \(p. 28\)](#).

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 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 chamador da API deve implementar lógica de novas tentativas para lidar com tempos limite do endpoint.

Esta demonstração complementa as instruções e os conceitos de [Criar uma API do API Gateway com integração à Lambda \(p. 28\)](#). Se você ainda não concluiu aquela demonstração, sugerimos que você faça isso primeiro.

Tópicos

- [Pré-requisitos \(p. 92\)](#)
- [Etapa 1: criar o recurso \(p. 92\)](#)
- [Etapa 2: criar o método GET \(p. 93\)](#)
- [Etapa 3: criar a função de execução do proxy de serviço da AWS \(p. 93\)](#)
- [Etapa 4: especificar configurações de método e testar o método \(p. 94\)](#)
- [Etapa 5: implantar a API \(p. 95\)](#)
- [Etapa 6: testar a API \(p. 95\)](#)
- [Etapa 7: Limpeza \(p. 96\)](#)

Pré-requisitos

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

1. Siga as etapas em [Pré-requisitos: prepare-se para compilar uma API no API Gateway \(p. 11\)](#).
2. Verifique se o usuário do IAM tem acesso para criar políticas e funções no IAM. Você precisa criar uma política e função IAM nesta demonstração.
3. Criar uma nova API chamada `MyDemoAPI`. Para obter mais informações, consulte [TUTORIAL: Criar uma API com integração não proxy HTTP \(p. 59\)](#).
4. Implante a API pelo menos uma vez em um estágio denominado `test`. Para obter mais informações, consulte [Implantar a API \(p. 43\)](#) em [Criar uma API do API Gateway com integração à Lambda \(p. 28\)](#).
5. Conclua as demais etapas em [Criar uma API do API Gateway com integração à Lambda \(p. 28\)](#).
6. 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 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. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Se `MyDemoAPI` for exibido, escolha Resources (Recursos).
3. No painel Resources (Recursos), escolha a raiz do recurso, representada por uma única barra invertida (/) e depois escolha Create Resource (Criar recurso).
4. Em Resource Name (Nome do recurso), digite `MyDemoAWSProxy` e, em seguida, selecione Create Resource (Criar recurso).

Etapa 2: 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.

Para criar o método GET

1. No painel Resources (Recursos), selecione /mydemoawsproxy e, em seguida, escolha Create Method (Criar método).
2. Para o método HTTP, escolha GET e depois salve sua opção.

Etapa 3: criar a função de execução do proxy de serviço da AWS

Nesta etapa, crie uma função do IAM que o seu proxy de serviço da AWS utiliza para interagir com o serviço da AWS. Chamamos essa função do IAM de função de execução de proxy de serviço da AWS. Sem ela, o API Gateway não pode interagir com o serviço da AWS. Nas etapas posteriores, você especifica essa função nas configurações do método GET que você acaba de criar.

Para criar a função de execução do proxy de serviço da AWS e sua política

1. Faça login no Console de gerenciamento da AWS e abra o console da IAM em <https://console.aws.amazon.com/iam/>.
2. Escolha Policies (Políticas).
3. Faça uma das coisas a seguir:
 - Se a página Welcome to Managed Policies (Bem-vindo às políticas gerenciadas) for exibida, escolha Get Started (Conceitos básicos) e escolha Create Policy (Criar política).
 - Se uma lista de políticas for exibida, escolha Create Policy (Criar política).
4. Próximo a Create Your Own Policy, escolha Select.
5. Em Policy Name (Nome da política), digite um nome para política (por exemplo, **APIGatewayAWSProxyExecPolicy**).
6. Em Description (Descrição), digite **Enables API Gateway to call AWS services**.
7. Para Policy Document, digite o seguinte e, então, selecione Create Policy.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Resource": [  
                "*"  
            ],  
            "Action": [  
                "sns>ListTopics"  
            ]  
        }  
    ]  
}
```

Esse documento de política permite que o chamador obtenha uma lista de tópicos do Amazon SNS para a conta da AWS.

8. Escolha Roles.

9. Selecione Create Role.
10. Escolha AWS Service (Serviço da AWS) em Select role type (Selecionar tipo de função) e, em seguida, escolha API Gateway.
11. Escolha Próximo: Permissões.
12. Selecione Next: Review.
13. Para Role Name (Nome da função), digite um nome para a função de execução (por exemplo, **APIGatewayAWSProxyExecRole**). Opcionalmente, digite uma descrição para a função e, em seguida, selecione Create role (Criar função).
14. Na lista Roles (Funções), escolha a função que você acaba de criar. Pode ser necessário rolar a lista para baixo.
15. Para a função selecionada, escolha Attach policy (Anexar política).
16. Marque a caixa de seleção ao lado da política criada anteriormente (por exemplo, **APIGatewayAWSProxyExecPolicy**) e selecione Attach policy (Anexar política).
17. A função que você acaba de criar tem a seguinte relação de confiança que permite à API Gateway assumir a função para quaisquer ações permitidas pelas políticas anexadas:

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

Para Role ARN (ARN da função), anote o nome de recurso da Amazon (ARN) para a função de execução. Você precisará disso mais tarde. O ARN deve ser semelhante a: `arn:aws:iam::123456789012:role/APIGatewayAWSProxyExecRole`, em que 123456789012 é o seu ID de conta da AWS.

Etapa 4: especificar configurações de método e testar o método

Nesta etapa, especifique as configurações para o método GET para que ele possa interagir com um serviço da AWS por meio do proxy de serviço da AWS. Em seguida, você testa o método.

Para especificar configurações para o método GET e testá-lo em seguida

1. No console do API Gateway, no painel Resources (Recursos) para a API denominada **MyDemoAPI**, em /mydemoawsproxy, escolha GET.
2. No painel Setup (Configuração), para Integration type (Tipo de integração), escolha Show advanced (Mostrar avançado) e escolha AWS Service Proxy (Proxy de serviço da AWS).
3. Para AWS Region (Região da AWS), escolha o nome da região da AWS onde você deseja obter os tópicos do Amazon SNS.
4. Em AWS Service (Serviço da AWS), selecione SNS.
5. Em HTTP method (Método HTTP), escolha GET.
6. Em Action, digite **ListTopics**.

7. Em Execution Role (Função de execução), digite o ARN para a função de execução.
8. Deixe Path Override (Sobreposição de caminho) em branco.
9. Escolha Salvar.
10. No painel Method Execution (Execução do método), na caixa Client (Cliente), selecione TEST e escolha Test (Testar). Se bem-sucedido, Response Body (Corpo da resposta) exibe uma resposta semelhante à 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 implanta a API para poder chamá-la de fora do console do API Gateway.

Para implantar a API

1. No painel Resources (Recursos), escolha Deploy API (Implantar API).
2. Em Deployment stage (Estágio de implantação), selecione test.
3. For Deployment description (Descrição da implantação), digite **Calling AWS service proxy walkthrough**.
4. Escolha Deploy (Implantar).

Etapa 6: testar a API

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

1. No painel Stage Editor (Editor de estágio), ao lado de Invoke URL (Invocar URL), copie a URL na área de transferência. A aparência deve ser semelhante a esta:

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

2. Cole a URL na caixa de endereço de uma nova guia do navegador.
3. Acrescente /mydemoawsproxy para obter 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

Você pode excluir os recursos do IAM necessários para o trabalho do proxy de serviço da AWS.

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 deixará de funcionar. A exclusão de um recurso do IAM não pode ser desfeita. Se quiser usar o recurso IAM novamente, você deverá recriá-lo.

Para 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, em seguida, escolha Role Actions (Ações da função), Delete Role (Excluir função). Quando solicitado, escolha Sim, excluir.
4. Na área Details (Detalhes), escolha Policies (Políticas).
5. Selecione APIGatewayAWSProxyExecPolicy e, em seguida, 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: Criar uma API REST como um proxy do Amazon S3 no API Gateway \(p. 116\)](#), [TUTORIAL: Criar uma API REST Calc com duas integrações de serviços da AWS e uma integração não proxy do Lambda \(p. 96\)](#) ou [TUTORIAL: Criar uma API REST como um proxy do Amazon Kinesis no API Gateway \(p. 142\)](#).

TUTORIAL: Criar uma API REST Calc com duas integrações de serviços da AWS e uma integração não proxy do Lambda

O Tutorial [conceitos básicos da integração não proxy \(p. 36\)](#) usa a integração do Lambda Function exclusivamente. A integração do 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 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 \(p. 112\)](#) para a API Calc, que você pode importar para o API Gateway seguindo as instruções em [the section called “Importar uma API REST” \(p. 356\)](#).

Tópicos

- [Crie uma conta do AWS \(p. 97\)](#)
- [Criar uma função do IAM assumível \(p. 97\)](#)
- [Crie uma função do Lambda Calc. \(p. 99\)](#)
- [Testar a função do Lambda Calc \(p. 99\)](#)
- [Criar uma API Calc \(p. 101\)](#)
- [Integração 1: Criar um método GET com parâmetros de consulta para chamar a função do Lambda \(p. 101\)](#)
- [Integração 2: Criar um método POST com uma carga JSON para chamar a função do Lambda \(p. 104\)](#)
- [Integração 3: Criar um método GET com parâmetros de caminho para chamar a função do Lambda \(p. 106\)](#)
- [Definições do OpenAPI da API de amostra integrada com uma função Lambda \(p. 112\)](#)

Crie uma conta do AWS

Antes de começar este tutorial, você precisará de uma conta da AWS.

Se você ainda não tiver uma conta da AWS, use o procedimento a seguir para criar uma.

Para cadastrar-se na AWS

1. Abra <https://aws.amazon.com/> e escolha Create an AWS Account.
2. Siga as instruções online.

Criar uma função do IAM assumível

Para que sua API invoque a função do Lambda Calc, será necessário ter uma função do IAM assumível do API Gateway, que é uma função do IAM com a seguinte relação de confiança:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "",  
            "Effect": "Allow",  
            "Action": "lambda:InvokeFunction",  
            "Resource": "arn:aws:lambda:us-east-1:123456789012:function:Calc"  
        }  
    ]  
}
```

```
    "Principal": {
        "Service": "apigateway.amazonaws.com"
    },
    "Action": "sts:AssumeRole"
}
]
```

A função que você criar precisará ter uma permissão do Lambda [InvokeFunction](#). Caso contrário, o chamador de API receberá uma resposta 500 Internal Server Error. Para conceder essa permissão à função, você poderá anexar a política do IAM a ela:

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

Veja a seguir como fazer isso:

Criar uma função do IAM assumível do API Gateway

1. Faça login no console do IAM.
2. Escolha Roles.
3. Selecione Create Role.
4. Em Select type of trusted entity, escolha Serviço da AWS.
5. Em Choose the service that will use this role, escolha Lambda.
6. Escolha Próximo: Permissões.
7. Escolha 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:

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

- b. Escolha Revisar política.
- c. Em Review Policy (Revisar política), faça o seguinte:
 - i. Em Name, digite um nome de usuário, como **lambda_execute**.
 - ii. Escolha 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:

```
{  
    "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.

Crie uma função do Lambda Calc.

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. Para Nome, digite **Calc**.
4. Defina Runtime (Tempo de execução) como Node.js 8.10.
5. Escolha Create function.
6. Copie o seguinte função Lambda e cole-a no editor de código do console do Lambda.

```
console.log('Loading the Calc function');  
  
exports.handler = function(event, context, callback) {  
    console.log('Received event:', JSON.stringify(event, null, 2));  
    if (event.a === undefined || event.b === undefined || event.op === undefined) {  
        callback("400 Invalid Input");  
    } else {  
        let result;  
        switch (event.op) {  
            case '+':  
                result = event.a + event.b;  
                break;  
            case '-':  
                result = event.a - event.b;  
                break;  
            case '*':  
                result = event.a * event.b;  
                break;  
            case '/':  
                result = event.a / event.b;  
                break;  
            default:  
                result = null;  
        }  
        callback(null, { result });  
    }  
};
```

```
}

var res = {};
res.a = Number(event.a);
res.b = Number(event.b);
res.op = event.op;

if (isNaN(event.a) || isNaN(event.b)) {
    callback("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":
        res.c = res.b==0 ? NaN : Number(event.a) / Number(event.b);
        break;
    default:
        callback("400 Invalid Operator");
        break;
}
callback(null, res);
};
```

7. Em Função de execução, selecione Choose an existing role (Selecionar uma função existente).
8. Insira o ARN da função para a função `lambda_invoke_function_assume_apigw_role` que você criou anteriormente.
9. Escolha Save (Salvar).

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"
}
```

Você deve 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 sua função Calc no console do Lambda:

1. No menu suspenso Saved test events (Eventos de teste salvos), selecione Configure test events (Configurar eventos de teste).
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. Selecione 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.

Criar a API Calc

1. No console API Gateway, escolha Create API.
2. Em API Name (Nome da API), digite **LambdaCalc**.
3. Deixe Description (Descrição) em branco e defina Endpoint Type (Tipo de endpoint) como Regional.
4. Escolha 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 repassa 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.

Para configurar o método **GET** com parâmetros de string de consulta

1. No console do API Gateway, em Resources (Recursos) da API `LambdaCalc`, selecione `/`.
2. No menu suspenso Actions (Ações), selecione Create Resource (Criar recurso).
3. Em Resource Name (Nome do recurso), insira `calc`.
4. Escolha Create Resource (Criar recurso).
5. Selecione o recurso `/calc` que você acabou de criar.
6. No menu suspenso Actions (Ações), escolha Create Method (Criar método).
7. No menu suspenso de métodos exibido, escolha GET.
8. Escolha o ícone de marca de seleção para salvar sua escolha.
9. No painel Set up (Configuração):
 - a. Em Integration type, escolha Serviço da AWS.
 - b. Em AWS Region (Região da AWS), escolha a região (por exemplo, `us-west-2`) em que você criou a função Lambda.
 - c. Para AWS Service, escolha Lambda.
 - d. Deixe AWS Subdomain (Subdomínio da AWS) em branco, pois a função Lambda não é hospedada em nenhum subdomínio da AWS.
 - e. Em HTTP method (método HTTP), escolha POST e escolha o ícone de marca de seleção para salvar sua opção. O Lambda requer que a solicitação `POST` seja usada para invocar qualquer função Lambda. Esse exemplo mostra que o método HTTP em uma solicitação de método de front-end pode ser diferente da solicitação de integração no back-end.
 - f. Escolha Use path override para Action Type. Essa opção nos permite especificar o ARN da ação `Invoke` para executar nossa função `calc`.
 - g. Insira `/2015-03-31/functions/arn:aws:lambda:region:account-id:function:Calc/invocations` em Path override (Substituição de caminho), em que `region` é a região na qual você criou sua função do Lambda e `account-id` é o número da conta para a conta da AWS.
 - h. Em Execution role (Função de execução), insira o ARN da função para a função `lambda_invoke_function_assume_apigw_role` do IAM que você criou anteriormente (p. 97).
 - i. Defina Content Handling (Manuseio de conteúdo) como Passthrough (Passagem), pois esse método não lidará com dados binários.
 - j. Deixe Use default timeout (Usar o tempo limite padrão) marcado.
 - k. Escolha Save (Salvar).
10. Escolha Method Request (Solicitação de método).

Agora você configurará parâmetros de consulta para o método GET em `/calc` para que ele possa receber a entrada em nome da função Lambda back-end.

- a. Escolha o ícone de lápis próximo a Request Validator (Validador de solicitação) e escolha Validar parâmetros de string de consulta e cabeçalhos no menu suspenso. Essa configuração fará com que uma mensagem de erro seja retornada, para indicar que os parâmetros necessários estão ausentes caso o cliente não os especifique. Você não será cobrado pela chamada ao back-end.
- b. Escolha o ícone de marca de seleção para salvar suas escolhas.
- c. Expanda a seção URL Query String Parameters (Parâmetros da string de consulta de URL).
- d. Escolha Add query string (Adicionar string de consulta).
- e. Para Nome, digite `operand1`.
- f. Escolha o ícone de marca de seleção para salvar o parâmetro.
- g. Repita as etapas anteriores para criar parâmetros denominados `operand2` e `operator`.
- h. Marque a opção Required para cada parâmetro para garantir que eles sejam validados.

11. Escolha Method Execution (Execução do método) e, em seguida, escolha Integration Request (Solicitação de integração) para configurar o modelo de mapeamento para converter as strings de consulta fornecidas pelo cliente na carga de solicitações de integração, conforme exigido pela função Calc.
 - a. Expanda a seção Mapping Templates (Modelos de mapeamento).
 - b. Escolha Quando nenhum modelo corresponde ao cabeçalho Content-Type para Request body passthrough (Passagem do corpo da solicitação).
 - c. Em Content-Type, escolha Add mapping template.
 - d. Digite **application/json** e escolha o ícone de marca de seleção para abrir o editor de modelo.
 - e. Escolha Yes, secure this integration para prosseguir.
 - f. Copie o seguinte script de mapeamento no editor de modelo de mapeamento:

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

Esse modelo mapeia os três parâmetros de consulta declarados em Method Request para valores de propriedade designados do objeto JSON como entrada para a função Lambda do back-end. O objeto JSON transformado será incluído como a carga útil da solicitação de integração.

- g. Escolha Save (Salvar).
12. Escolha Method Execution (Execução de método).
13. Agora, você pode testar o método GET para verificar se ele foi configurado corretamente para invocar a função Lambda:
 - a. Em Query Strings (Strings de consulta), digite **operand1=2&operand2=3&operator=+**.
 - b. Selecione Test (Testar).

Os resultados devem ser semelhantes ao seguinte:

Request: /calc?operand1=2&operand2=3&operator=+

Status: 200

Latency: 414 ms

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 back-end no corpo da solicitação. Para garantir que o cliente faça upload dos dados de entrada corretos, você habilitará a validação de solicitações na carga.

Para configurar o método POST com uma carga JSON para invocar uma função do Lambda

1. Acesse o console do API Gateway.
2. Escolha APIs.
3. Selecione a API LambdaCalc criada anteriormente.
4. Escolha o recurso /calc no painel Resources.
5. No menu Actions (Ações), escolha Create Method (Criar método).
6. Escolha POST na lista suspensa de métodos.
7. Escolha o ícone de marca de seleção para salvar sua escolha.
8. No painel Set up (Configuração):
 - a. Em Integration type, escolha Serviço da AWS.
 - b. Em AWS Region (Região da AWS), escolha a região (por exemplo, us-west-2) em que você criou a função Lambda.
 - c. Para AWS Service, escolha Lambda.
 - d. Deixe AWS Subdomain (Subdomínio da AWS) em branco, pois a função Lambda não é hospedada em nenhum subdomínio da AWS.
 - e. Em Método HTTP, escolha POST. Esse exemplo mostra que o método HTTP em uma solicitação de método de front-end pode ser diferente da solicitação de integração no back-end.
 - f. Em Action (Ação), selecione Use path override para Action Type (Tipo de ação). Essa opção permite especificar o ARN da ação **Invoke** para executar a sua função Calc.
 - g. Em Path override (Substituição de caminho), digite /2015-03-31/functions/
`arn:aws:lambda:region:account-id:function:Calc/invocations`, em que *region* é a região na qual você criou sua função do Lambda e *account-id* é o número da conta para a conta da AWS.
 - h. Em Execution role (Função de execução), insira o ARN da função para a função **lambda_invoke_function_assume_apigw_role** do IAM que você criou anteriormente (p. 97).
 - i. Defina Content Handling (Manuseio de conteúdo) como Passthrough (Passagem), pois esse método não lidará com dados binários.
 - j. Deixe Use Default Timeout (Usar o tempo limite padrão) marcado.
 - k. Escolha Save (Salvar).
9. Selecione Models (Modelos) na API LambdaCalc do painel de navegação principal do console do API Gateway para criar modelos de dados para a entrada e a saída do método:
 - a. Escolha Create no painel Models. Digite Input em Model name (Nome do modelo), digite application/json em Content type (Tipo de conteúdo) e copie a seguinte definição de esquema na caixa Model schema (Esquema de modelo):

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

```
        },
        "title":"Input"
    }
```

Esse modelo descreve a estrutura de dados de entrada e será usado para validar o corpo da solicitação recebida.

- b. Escolha Create model (Criar modelo).
- c. Escolha Create no painel Models. Digite Output em Model name (Nome do modelo), digite application/json em Content type (Tipo de conteúdo) e copie a seguinte definição de esquema na caixa Model schema (Esquema de modelo):

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

Esse modelo descreve a estrutura de dados da saída calculada do back-end. 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.

- d. Localize o ID da API para a sua API na parte superior da tela do console e o anote. Ele será exibido entre parênteses após o nome da API.
- e. No painel Models (Modelos), digite Create (Criar).
- f. Digite Result em Model name (Nome do modelo).
- g. Digite application/json em Content type (Tipo de conteúdo).
- h. Copie a seguinte definição de esquema, em que *restapi-id* é o ID da API REST que você anotou anteriormente, na caixa Model schema (Esquema do modelo):

```
{
    "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":"Output"
}
```

Esse modelo descreve a estrutura de dados dos dados de resposta retornados. Ele faz referência aos esquemas Input e Output definidos na API especificada (*restapi-id*). Novamente, esse modelo não é usado neste tutorial, pois ele tira proveito do comportamento de passagem direta.

- i. Escolha Create model (Criar modelo).
10. No painel de navegação principal, na sua API LambdaCalc, selecione Resources (Recursos).
11. No painel Resources (Recursos), escolha o método POST para sua API.
12. Escolha Method Request (Solicitação de método).
13. Nas configurações de Method Request, faça o seguinte para habilitar a validação de solicitações no corpo da solicitação recebida:

- a. Escolha o ícone de lápis próximo a Request Validator para escolher validate body. Escolha o ícone de marca de seleção para salvar sua escolha.
 - b. Expanda a seção Request Body, escolha Add model
 - c. Digite application/json no campo de entrada Content-Type e escolha Input na lista suspensa na coluna Model name. Escolha o ícone de marca de seleção para salvar sua escolha.
14. Para testar seu método POST, faça o seguinte:
- a. Escolha Method Execution (Execução de método).
 - b. Selecione Test (Testar).
15. Copie a seguinte carga JSON no Request Body (Corpo da solicitação):

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

16. Selecione Test (Testar).

Você deverá ver a saída a seguir no Response Body (Corpo da resposta):

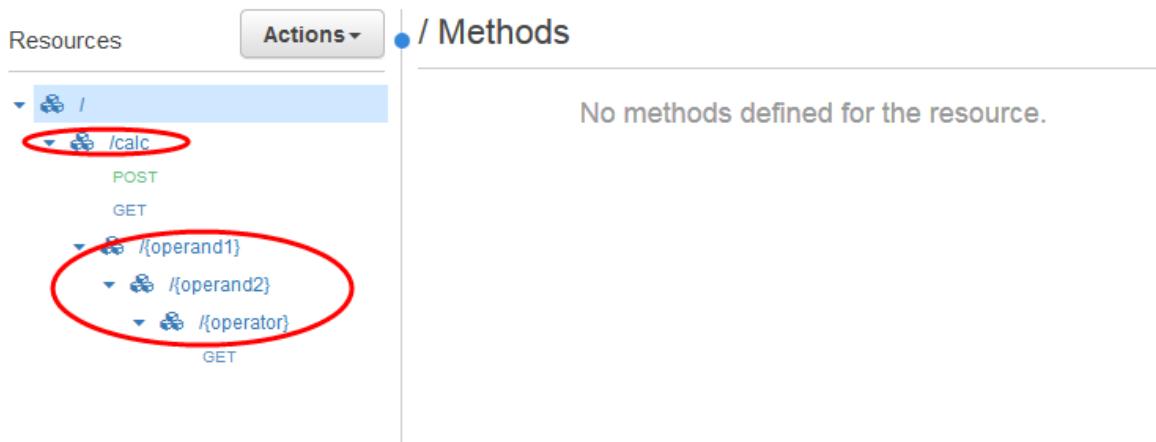
```
{  
    "a": 1,  
    "b": 2,  
    "op": "+",  
    "c": 3  
}
```

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 do back-end. Os valores de parâmetros de caminho especificam os dados de entrada para a função 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.

Dessa vez, você usará o suporte de integração do Lambda incorporado no console do API Gateway para configurar a integração do método.

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



Para configurar um método **GET** com parâmetros de caminho de URL

1. Acesse o console do API Gateway.
2. Em APIs, selecione a API LambdaCalc criada anteriormente.
3. No painel de navegação Recursos de API, escolha /calc.
4. No menu suspenso Actions, escolha Create Resource.
5. Em Resource Name (Nome do recurso), digite **{operand1}**.
6. Em Resource Path (Caminho do recurso), digite **{operand1}**.
7. Escolha Create Resource (Criar recurso).
8. Escolha o recurso /calc/{operand1} que você acabou de criar.
9. No menu suspenso Actions, escolha Create Resource.
10. Em Resource Name (Nome do recurso), digite **{operand2}**.
11. Em Resource Path (Caminho do recurso), digite **{operand2}**.
12. Escolha Create Resource (Criar recurso).
13. Escolha o recurso /calc/{operand1}/{operand2} que você acabou de criar.
14. No menu suspenso Actions, escolha Create Resource.
15. Em Resource Path (Caminho do recurso), digite **{operator}**.
16. Em Resource Name (Nome do recurso), digite **{operator}**.
17. Escolha Create Resource (Criar recurso).
18. Escolha o recurso /calc/{operand1}/{operand2}/{operator} que você acabou de criar.
19. No menu suspenso Actions (Ações), escolha Create Method (Criar método).
20. No menu suspenso de métodos, escolha **GET**.
21. No painel Configuração, escolha **Lambda Function** como Tipo de integração para usar o processo de configuração simplificado habilitado pelo console.
22. Escolha uma região (por exemplo, **us-west-2**) para Lambda Region. Esta é a região em que a função Lambda está hospedada.
23. Escolha sua função Lambda existente (**Calc**) para Lambda Function.
24. Escolha Save e, então, escolha OK para consentir em Add Permissions to Lambda Function.
25. Escolha Integration Request (Solicitação de integração).
26. Configure o modelo de mapeamento da seguinte maneira:
 - a. Expanda a seção **Mapping Templates** (Modelos de mapeamento).
 - b. Deixe definido como **When no template matches the requested Content-Type header** (Quando nenhum modelo corresponder ao cabeçalho Content-Type solicitado).

- c. Escolha Add mapping template (Adicionar modelo de mapeamento).
- d. Digite application/json para Content-Type e escolha o ícone de marca de seleção para abrir o editor de modelo.
- e. Escolha Yes, secure this integration para prosseguir.
- f. Copie o seguinte script de mapeamento no editor de modelo:

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

Esse modelo mapeia os três parâmetros de caminho de URL, declarados quando o recurso /calc/{operand1}/{operand2}/{operator} foi criado, para 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 %2F em ' / ' antes de repassá-lo para a função do Lambda.

- g. Escolha Save (Salvar).

Quando o método está configurado corretamente, as configurações devem ser semelhantes às seguintes:

◀ Method Execution /calc/{operand1}/{operand2}/{operator} - GET - Integration...

Provide information about the target backend that this method will call and whether the incoming request data should be modified.

Integration type Lambda Function 

HTTP 
 Mock 
 AWS Service 

Use Lambda Proxy integration 

Lambda Region us-west-2 

Lambda Function Calc 

Invoke with caller credentials 

Credentials cache Do not add caller credentials to cache key 

▼ Body Mapping Templates

Request body passthrough When no template matches the request Content-Type header 
 When there are no templates defined (recommended) 
 Never 

Content-Type
application/json

 Add mapping template

application/json

Generate template: 

```
1 ~ {  
2   "a": "$input.params('operand1')",  
3   "b": "$input.params('operand2')",  
4   "op": "#if($input.params('operator')=='%2F')/#{else}{$input.params('operator')}#end"  
5 }  
6 }
```

27. Para testar a função GET, faça o seguinte:

- a. Escolha Method Execution (Execução de método).
- b. Selecione Test (Testar).
- c. Digite 1, 2 e + nos campos {operand1}, {operand2} e {operator}, respectivamente.
- d. Selecione Test (Testar).
- e. O resultado deve ser semelhante ao seguinte:

Request: /calc/1/1/+

Status: 200

Latency: 816 ms

Response Body

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

Response Headers

```
{"X-Amzn-Trace-Id":"sampled=0;root=1-58f7f1f6-57c26581ed7073a711c13216","Content-Type":"application/json"}
```

Logs

```
Execution log for request test-request  
Wed Apr 19 23:25:42 UTC 2017 : Starting execution for request: test-invoke-request  
Wed Apr 19 23:25:42 UTC 2017 : HTTP Method: GET, Resource Path: /calc/1/1/+  
Wed Apr 19 23:25:42 UTC 2017 : Method request path: {operand1=1, operand2=1, operator =+}  
Wed Apr 19 23:25:42 UTC 2017 : Method request query string: {}  
Wed Apr 19 23:25:42 UTC 2017 : Method request headers: {}  
Wed Apr 19 23:25:42 UTC 2017 : Method request body before transformations:  
Wed Apr 19 23:25:42 UTC 2017 : Endpoint request URI: https://lambda.us-west-2.amazonaws.com/2015-03-31/functions/arn:aws:lambda:us-west-2:738575810317:function:Calc/invocations  
Wed Apr 19 23:25:42 UTC 2017 : Endpoint request headers: {x-amzn-lambda-integration-tag=test-request, Authorization=*****}
```

Esse resultado de teste mostra a saída original da função do Lambda do back-end, conforme repassada por meio da resposta de integração sem mapeamento, pois não há um modelo de mapeamento. Você modelará a estrutura de dados da carga de resposta de método após o esquema Result.

28. 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.
- Seleciona /calc/{operand1}/{operand2}/{operator}.
 - Seleciona GET.
 - Escolha Method Execution (Execução de método).
 - Seleciona Method Response (Resposta de método).

- e. Expanda a resposta 200,
- f. Em Response Body for 200, escolha o ícone de lápis ao lado do modelo para o tipo de conteúdo application/json.
- g. Na lista suspensa Modelos, escolha Result.
- h. Escolha o ícone de marca de seleção para salvar sua escolha.

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.

29. Para criar o modelo de mapeamento, faça o seguinte:
 - a. Escolha Method Execution (Execução de método).
 - b. Escolha Integration Response e expanda a entrada de resposta do método 200.
 - c. Expanda a seção Mapping Templates (Modelos de mapeamento).
 - d. Escolha application/json na lista Content-Type.
 - e. Escolha Result na lista suspensa Generate template para ativar o blueprint de modelo Result.
 - f. Altere o blueprint de modelo para o seguinte:

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

- g. Escolha Save (Salvar).
30. Para testar o modelo de mapeamento, faça o seguinte:
 - a. Escolha Method Execution (Execução de método).
 - b. Selecione Test (Testar).
 - c. Digite 1 2 e + nos campos de entrada operand1, operand2 e operator, respectivamente.

A resposta de integração da função do Lambda agora é mapeada para um objeto Result:

- d. Selecione Test (Testar) e você verá o seguinte em Response Body (Corpo de resposta) no console:

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

31. Neste momento, a API pode ser chamada apenas por meio de Test Invoke (Invocação de teste) no console do API Gateway. Para disponibilizá-la para os clientes, você precisará implantá-la da seguinte forma:

- a. Escolha Deploy API (Implantar API) no menu suspenso Actions (Ações).
- b. Selecione [New Stage] ([Novo estágio]) no menu suspenso Deployment Stage (Estágio de implantação).
- c. Em Stage Name (Nome do estágio), insira **test**.
- d. Escolha Deploy (Implantar).
- e. Anote o Invoke URL (URL de invocação) na parte superior da janela do console. Você pode usá-lo com ferramentas, como [Postman](#) e [cURL](#) para testar sua API.

Note

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 recursos ou atualizações não estarão disponíveis para clientes de sua API.

Definições do OpenAPI da API de amostra integrada com uma função Lambda

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "version": "2017-04-20T04:08:08Z",
    "title": "LambdaCalc"
  },
  "host": "uojnr9hd57.execute-api.us-east-1.amazonaws.com",
  "basePath": "/test",
  "schemes": [
    "https"
  ],
  "paths": {
    "/calc": {
      "get": {
        "consumes": [
          "application/json"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "operand2",
            "in": "query",
            "required": true,
            "type": "string"
          },
          {
            "name": "operator",
            "in": "query",
            "required": true,
            "type": "string"
          },
          {
            "name": "operand1",
            "in": "query",
            "required": true,
            "type": "string"
          }
        ]
      }
    }
  }
}
```

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

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

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

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

TUTORIAL: Criar uma API REST como um proxy do Amazon S3 no API Gateway

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 chamador](#).
- Expor GET em um recurso Folder para [ver 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 [ver ou fazer download de 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](#).

Note

Para integrar sua API do API Gateway ao Amazon S3, você deve escolher uma região em que tanto os serviços do API Gateway quanto do Amazon S3 estejam disponíveis. Para conhecer a disponibilidade das regiões, consulte [Regiões e endpoints](#).

Você pode querer importar a API de amostra como um proxy Amazon S3, conforme mostrado em [Definições do OpenAPI da API de exemplo como um proxy do Amazon S3 \(p. 133\)](#). Para obter instruções sobre como importar uma API usando a definição do OpenAPI, consulte [Importar uma API REST no API Gateway \(p. 356\)](#).

Para usar o console do API Gateway para criar a API, você deve primeiro se cadastrar para uma conta da AWS.

Se você ainda não tiver uma conta da AWS, use o procedimento a seguir para criar uma.

Para cadastrar-se na AWS

1. Abra <https://aws.amazon.com/> e escolha Create an AWS Account.
2. Siga as instruções online.

Tópicos

- [Configurar permissões do IAM para a API invocar ações do Amazon S3 \(p. 117\)](#)
- [Criar recursos de API para representar recursos do Amazon S3 \(p. 118\)](#)
- [Exponha um método de API para listar os buckets do Amazon S3 do chamador \(p. 119\)](#)
- [Exportar métodos de API para acessar um bucket do Amazon S3 \(p. 125\)](#)
- [Exportar métodos de API para acessar um objeto do Amazon S3 em um bucket \(p. 128\)](#)
- [Chamar a API usando um cliente de API REST \(p. 131\)](#)
- [Definições do OpenAPI da API de exemplo como um proxy do Amazon S3 \(p. 133\)](#)

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

Para permitir que a API invoque ações necessárias do Amazon S3, você deve ter as políticas IAM apropriadas anexadas a uma função IAM. A próxima seção descreve como verificar e criar, se necessário, as políticas e a função IAM necessárias.

Para que a sua API visualize ou liste buckets e objetos do Amazon S3, você pode usar a política AmazonS3ReadOnlyAccess fornecida pelo IAM na função IAM. O ARN dessa política é `arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess`, conforme mostrado a seguir:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3:Get*",  
                "s3>List*"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

Esse documento de política afirma que qualquer uma das ações `Get*` e `List*` do Amazon S3 pode ser invocada em qualquer um dos recursos do Amazon S3.

Para que a sua API atualize buckets e objetos do Amazon S3, você pode usar uma política personalizada para qualquer uma das ações `Put*` do Amazon S3, conforme mostrado a seguir:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "s3:Put*",  
            "Resource": "*"  
        }  
    ]  
}
```

```
}
```

Para que a sua API funcione com ações `Get*`, `List*` e `Put*` do Amazon S3, você pode adicionar as políticas somente leitura e somente inserção acima à função IAM.

Para que a sua API invoque as ações `Post*` do Amazon S3, você deve usar uma política de permissão para as ações `s3:Post*` na função IAM. Para obter uma lista completa de ações do Amazon S3, consulte [Especificando permissões do Amazon S3 em uma política](#).

Para que sua API crie, visualize, atualize e exclua buckets e objetos no Amazon S3, você pode usar a política `AmazonS3FullAccess` fornecida pelo IAM na função IAM. O ARN é `arn:aws:iam::aws:policy/AmazonS3FullAccess`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": "*"
        }
    ]
}
```

Depois de escolher as políticas IAM desejadas para uso, crie uma função IAM e anexe as políticas a ela. A função IAM resultante deve conter a seguinte política de confiança para o API Gateway assumir essa função em tempo de execução.

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

Ao usar o console do IAM para criar a função, escolha o tipo de função Amazon API Gateway para garantir que essa política de confiança seja automaticamente incluída.

Criar recursos de API para representar recursos do Amazon S3

Usaremos o recurso da raiz da API (/) como o contêiner de buckets do Amazon S3 de um chamador autenticado. Também criaremos 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 chamador.

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`.

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

1. No console do API Gateway, crie uma API denominada MyS3. O recurso raiz dessa API (/) representa o serviço Amazon S3.
2. No recurso raiz da API, crie um recurso filho denominado Pasta e defina o Caminho do recurso necessário como `/{folder}`.
3. Para o recurso Pasta da API, crie um recurso filho Item. Defina o Caminho do recurso como `/{item}`.

The screenshot shows the 'New Child Resource' dialog box. On the left, the 'Resources' sidebar shows a tree structure with a node labeled `/{folder}` highlighted with a red oval. The main form has the following fields:

- Configure as proxy resource:** An unchecked checkbox with a help icon.
- Resource Name***: A text input field containing the value `Item`.
- Resource Path***: A text input field containing the value `/{folder}/ {item}`, which is also highlighted with a red oval.
- Enable API Gateway CORS:** An unchecked checkbox with a help icon.

At the bottom right, there are 'Cancel' and 'Create Resource' buttons, with 'Create Resource' also highlighted with a red oval.

Exponha um método de API para listar os buckets do Amazon S3 do chamador

Obter a lista de buckets do Amazon S3 do chamador envolve invocar a ação [GET Service](#) no Amazon S3. No recurso raiz do API, (/), crie o método GET. Configure o método GET para integrar o Amazon S3, da seguinte maneira.

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

1. Escolha Criar método no nó raiz (/) no menu suspenso Ações, no canto superior direito do painel Recursos.
2. Escolha GET na lista suspensa de verbos HTTP e escolha o ícone de marca de seleção para começar a criar o método.

The screenshot shows the '/ Methods' list for the root resource. The left sidebar shows the tree structure with a node labeled `/{folder}` expanded, showing the child node `/{item}`. The main area displays the message: "No methods defined for the resource." There is a 'Create Method' button at the top right of the list area.

3. No painel / - GET – Configuração, escolha Serviço da AWS como Tipo de integração.
4. Na lista, escolha uma região (por exemplo, us-west-2) para Região da AWS
5. Em Serviço da AWS, escolha S3.
6. Deixe o campo Subdomínio da AWS em branco.
7. Em Método HTTP, escolha GET.
8. Em Tipo de ação, escolha Usar sobreposição 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 s3-host-name e passa o cliente especificado bucket e key do cliente para o Amazon S3.
9. (Opcional) Em Sobreposição de caminho digite /.
10. Copie o ARN da função IAM criada anteriormente (no console do IAM) e cole-o em Função de execução.
11. Deixe as outras configurações como o padrão.
12. Escolha Salvar para concluir a configuração desse método.

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

Note

Após a configuração inicial, você pode modificar essas configurações na página Solicitação de integração do método.

Para controlar quem pode chamar esse método da nossa API, podemos ativar o sinalizador de autorização de método e defini-lo como AWS_IAM.

Para permitir que o IAM controle o acesso ao método o GET /

1. Na Execução de método, escolha Solicitação de método.
2. Escolha o ícone de lápis ao lado de Autorização
3. Escolha AWS_IAM na lista suspensa.
4. Escolha o ícone de marca de seleção para salvar a configuração.

[Method Execution](#) / - GET - Method Request

Provide information about this method's authorization settings and the parameters it can receive.

Authorization Settings

Authorization AWS_IAM 

API Key Required false 

▶ URL Query String Parameters

▶ HTTP Request Headers

▶ Request Models [Create a Model](#)

Para que a nossa API retorne respostas e exceções bem-sucedidas corretamente ao chamador, declaramos as respostas 200, 400 e 500 em Resposta de método. Usamos 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.

Para declarar tipos de resposta para o método GET /

1. No painel Execução de método, escolha a caixa Resposta de método. O API Gateway declara a resposta 200 por padrão.
2. Escolha Adicionar resposta, digite **400** na caixa de texto de entrada e escolha a marca de seleção para terminar a declaração.
3. Repita a etapa acima para declarar o tipo de resposta 500. A configuração final é mostrada da seguinte maneira:

[← Method Execution](#) / - GET - Method Response

Provide information about this method's response types, their headers and content types.

	HTTP Status	
▶	200	✖
▶	400	✖
▶	500	✖
+ Add Response		

Como a resposta de integração bem-sucedida do Amazon S3 retorna a lista de buckets como uma carga XML, e a resposta de método padrão do API Gateway retorna uma carga JSON, devemos 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, também queremos mostrar para o cliente outros parâmetros de cabeçalho, como Date e Content-Length.

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

1. No console do API Gateway, escolha Resposta de método. Adicione o cabeçalho Content-Type ao tipo de resposta 200.

[Method Execution](#) / - GET - Method Response

Provide information about this method's response types, their headers and content types.

HTTP Status	
▼	200 

Response Headers for 200	
Name	
Timestamp	 
Content-Length	 
Content-Type	 
+ Add Header	

Response Models for 200	
Content type	Models
application/json	Empty  
+ Add Response Model	

▶ 400 
▶ 500 

[+ Add Response](#)

2. Em Resposta de integração, Content-Type, digite `integration.response.header.Content-Type` para o método resposta.

[Method Execution](#) / - GET - Integration Response

First, declare response types using [Method Response](#). Then, map the possible responses from the backend to this method's response types.

	HTTP status regex	Method response status	Output model	Default mapping	
▼	-	200		Yes	✖

Map the output from your HTTP endpoint to the headers and output model of the 200 method response.

HTTP status regex ⓘ

Method response status

[Cancel](#) [Save](#)

▼ Header Mappings

Response header	Mapping value ⓘ	
Timestamp	integration.response.header.Date	✎
Content-Length	integration.response.header.Content-Length	✎
Content-Type	integration.response.header.Content-Type	✎

▶ Body Mapping Templates

Com os mapeamentos de cabeçalho acima, o API Gateway converte o cabeçalho **Date** do back-end no cabeçalho **Timestamp** para o cliente.

3. Ainda na Resposta de integração, escolha Adicionar integração apropriada, digite uma expressão regular na caixa de textoRegex de status HTTP para um status de resposta de método restante. Repita até que todos os status de respostas de método sejam cobertos.

[Method Execution](#) / - GET - Integration Response

First, declare response types using [Method Response](#). Then, map the possible responses from the backend to this method's response types.

	HTTP status regex	Method response status	Output model	Default mapping	
▶	-	200		Yes	✖
▶	4\d{2}	400		No	✖
▼	5\d{2}	500		No	✖

Map the output from your HTTP endpoint to the headers and output model of the 500 method response.

HTTP status regex 5\d{2} i

Method response status 500

Cancel **Save**

▼ Header Mappings

Response header	Mapping value i
No method response headers.	

▶ Body Mapping Templates

Add integration response +

Como prática recomendada, vamos testar a API que configuramos até agora.

Testar o método GET no recurso raiz da API

1. Volte para Execução de método e escolha Testar na caixa Cliente.
2. Escolha Testar no painel GET/ - Teste de método. Um exemplo de resultado é mostrado da seguinte forma.

Note

Para usar o console do API Gateway para testar a API como um proxy do Amazon S3, certifique-se de que o bucket do S3 desejado seja de uma região diferente da região da API. Caso contrário, você poderá receber uma resposta de erro interno de servidor 500. Essa limitação não se aplica a nenhuma API implantada.

Expor métodos de API para acessar um bucket do Amazon S3

Para trabalhar com um bucket do Amazon S3, expomos os métodos GET, PUT e DELETE no recurso /{folder} para listar objetos em um bucket, criar um novo bucket e excluir um bucket existente. As instruções são semelhantes às descritas em [Exponha um método de API para listar os buckets do Amazon S3 do chamador \(p. 119\)](#). Nas discussões seguintes, descrevemos as tarefas gerais e realçamos as diferenças relevantes.

Para expor métodos GET, PUT e DELETE em um recurso de pasta

1. No nó `/ffolder` da árvore Recursos, crie os métodos DELETE, GET e PUT, um de cada vez.

- Configure a integração inicial de cada método criado com os endpoints correspondentes do Amazon S3. A captura de tela a seguir ilustra essa configuração para o método PUT /{folder}. Para os métodos DELETE /{folder} e GET /{folder}, substitua o valor PUT de Método HTTP por DELETE e GET, respectivamente.

The screenshot shows the AWS Lambda Integration configuration page for the PUT /{folder} method. The left sidebar lists methods: GET, DELETE, GET, PUT, and /{item}. The main area shows the integration configuration:

- Integration type:** AWS Service (selected)
- AWS Region:** <region>
- AWS Service:** S3 (highlighted with a red circle)
- AWS Subdomain:** (empty)
- HTTP method:** PUT (highlighted with a red circle)
- Action Type:** Use path override (selected)
- Path override (optional):** {bucket} (highlighted with a red circle)
- Execution role:** arn:aws:iam:::role/apigAwsProxyRole
- Save** button

Observe que usamos o parâmetro de caminho {bucket} nas URLs do endpoint do Amazon S3 para especificar o bucket. Precisaremos mapear o parâmetro de caminho {folder} das solicitações de método para o parâmetro de caminho {bucket} das solicitações de integração.

- Para mapear {folder} para {bucket}:
 - Escolha Execução de método e Solicitação de integração.
 - Expanda Parâmetros de caminho de URL e escolha Adicionar caminho
 - Digite bucket na coluna Nome e method.request.path.folder na coluna Mapeado de. Escolha o ícone de marca de seleção para salvar o mapeamento.

▼ URL Path Parameters

Name	Mapped from	Caching
bucket	method.request.path.folder	<input checked="" type="checkbox"/>

- Em Solicitação de método, adicione Content-Type à seção Cabeçalhos de solicitação HTTP.

▼ HTTP Request Headers

Name	Caching
Content-Type	<input type="checkbox"/>
+ Add header	

Isso é necessário principalmente para testar, ao usar o console do API Gateway, quando você deve especificar **application/xml** para uma carga XML.

5. Em Solicitação de integração, configure os seguintes mapeamentos de cabeçalho, seguindo as instruções descritas em [Exponha um método de API para listar os buckets do Amazon S3 do chamador \(p. 119\)](#).

▼ HTTP Headers

Name	Mapped from ⓘ	Caching
x-amz-acl	'authenticated-read'	<input type="checkbox"/>  
Content-Type	method.request.header.Content-Type	<input type="checkbox"/>  
+ Add header		

O cabeçalho **x-amz-acl** é para especificar o controle de acesso na pasta (ou o bucket do Amazon S3 correspondente). Para obter mais informações, consulte [Solicitação do bucket PUT do Amazon S3](#).

6. Para testar o método **PUT**, escolha Testar na caixa Cliente de Execução de método e digite o seguinte como entrada para o teste:
 - a. Em pasta, digite um nome de bucket,
 - b. Para o cabeçalho Content-Type, digite **application/xml**.
 - c. Em Corpo da solicitação, forneça a região do bucket como a restrição de localização, declarada em um fragmento de XML como a carga da solicitação. Por exemplo,

```
<CreateBucketConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <LocationConstraint>{region}</LocationConstraint>
</CreateBucketConfiguration>
```

The screenshot shows the AWS Lambda Test API interface. A red circle highlights the 'Path' field containing 'my-pictures-02-16-2016'. Another red circle highlights the 'Content-Type' header field set to 'application/xml'. A third red circle highlights the 'Request Body' field, which contains XML code for creating a bucket:

```
<CreateBucketConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01">
  <LocationConstraint>us-west-2</LocationConstraint>
</CreateBucketConfiguration>
```

The right side of the interface displays the request details: Request: /my-pictures-02-16-2016, Status: 200, Latency: 984 ms, and Response Body: no data. The Response Headers show Content-Length: 0 and Content-Type: application/json. The Logs section shows the execution log for the request.

7. Repita as etapas anteriores para criar e configurar o método GET e DELETE no recurso /{folder} da API.

Os exemplos acima ilustram como criar um novo bucket na região especificada, ver a lista de objetos no bucket e excluir o bucket. Outras operações de bucket do Amazon S3 permitem trabalhar com metadados ou propriedades do bucket. Por exemplo, você pode configurar sua API para chamar a ação [PUT /?notification](#) do Amazon S3 para configurar notificações no bucket ou chamar [PUT /?acl](#) para definir uma lista de controle de acesso no bucket, entre outros. A configuração da API é semelhante, exceto pelo fato de que você precisa anexar parâmetros de consulta apropriados às URLs do endpoint do Amazon S3. Em tempo de execução, você deve fornecer a carga XML apropriada para a solicitação de método. O mesmo pode ser dito sobre o suporte às outras operações GET e DELETE em um bucket do Amazon S3. Para obter mais informações sobre possíveis ações do S3 em um bucket, consulte [Operações em buckets do Amazon S3](#).

Exportar 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. Para obter uma lista completa de ações com suporte, consulte [Operações do Amazon S3 em objetos](#).

Neste tutorial, expomos a operação [PUT Object](#), a operação [GET Object](#), a operação [HEAD Object](#) e a operação [DELETE Object](#) por meio dos métodos de API de `PUT /{folder}/{item}`, `GET /{folder}/{item}`, `HEAD /{folder}/{item}` e `DELETE /{folder}/{item}`, respectivamente.

As configurações de API para os métodos PUT, GET e DELETE em `/ {folder} / {item}` são semelhantes àquelas em `/ {folder}`, conforme prescrito em [Exportar métodos de API para acessar um bucket do Amazon S3 \(p. 125\)](#). Uma diferença significativa é que caminho de solicitação relacionado ao objeto tem um parâmetro de caminho adicional de `{item}` e este parâmetro de caminho deve ser mapeado para o parâmetro de caminho de solicitação de integração `{object}`.

[Method Execution](#) /{folder}/{item} - PUT - Integration Request

Provide information about the target backend that this method will call and whether the incoming request data should be modified.

Integration type Lambda Function [i](#)

HTTP [i](#)

Mock [i](#)

AWS Service [i](#)

AWS Region [u](#) 2 [e](#)

AWS Service S3 [e](#)

AWS Subdomain [e](#)

HTTP method PUT [e](#)

Path override {bucket}/{object} [e](#)

Execution role arn:aws:iam::[...](#):role/apigAwsProxyRole [e](#)

Credentials cache Do not add caller credentials to cache key [e](#)

▼ URL Path Parameters

Name	Mapped from i	Caching	e
object	method.request.path.item	<input type="checkbox"/>	e
bucket	method.request.path.folder	<input type="checkbox"/>	e

[+ Add path](#)

O mesmo é válido para os métodos GET e DELETE.

Como ilustração, a captura de tela a seguir mostra a saída ao testar o método GET em um recurso {folder}/{item} usando o console do API Gateway. A solicitação retorna corretamente o texto sem formatação ("Welcome to README.txt") como o conteúdo do arquivo especificado (README.txt) no bucket do Amazon S3 especificado (apig-demo).

← Method Execution /{folder}/{item} - GET - Method Test

Make a test call to your method with the provided input

Path

folder

apig-demo

item

README.txt

Request: /apig-demo/README.txt

Status: 200

Latency: 486 ms

Response Body

Welcome to README.txt

Response Headers

{"Content-Type": "text/plain"}

Logs

```
Execution log for request test-request
Fri Feb 19 03:35:20 UTC 2016 : Starting execution for request: test-invoke-request
Fri Feb 19 03:35:20 UTC 2016 : API Key: test-invoke-api-key
Fri Feb 19 03:35:20 UTC 2016 : Method request path: {item=README.txt, folder=apig-demo}
Fri Feb 19 03:35:20 UTC 2016 : Method request query string: {}
Fri Feb 19 03:35:20 UTC 2016 : Method request headers: {}
Fri Feb 19 03:35:20 UTC 2016 : Method request body before transformations: null
Fri Feb 19 03:35:20 UTC 2016 : Endpoint request URI: http://s3-us-west-2.amazonaws.com/apig-demo/README.txt
Fri Feb 19 03:35:20 UTC 2016 : Endpoint request headers: {Authorization=*****}
*****
*****
*****
*****
*****a99006, X-Amz
```

Query Strings

No query string parameters exist for this method.
You can add them via Method Request.

Headers

No header parameters exist for this method. You can add them via Method Request.

Stage Variables

No stage variables exist for this method.

Client Certificate

None

Request Body

Request Body is not supported for GET methods.

Test

Para fazer download ou upload de arquivos binários, que no API Gateway é qualquer conteúdo que não seja JSON codificado em utf-8, as configurações de API adicionais são necessárias. 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. Escolha Suporte binário para a API (no painel de navegação principal do API Gateway).
 - b. Selecione Edit.
 - c. Digite o tipo de mídia necessário (por exemplo, image/png em Tipos de mídia binários).
 - d. Escolha Adicionar tipo de mídia binário 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 [Comportamentos de passagem direta de integração \(p. 304\)](#) e [Selecionar um modelo de mapeamento VTL \(p. 303\)](#).

O tamanho máximo da carga é 10 MB. Consulte [Limites do API Gateway para configurar e executar uma API REST \(p. 726\)](#).

Certifique-se de que os arquivos no Amazon S3 têm 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 \(p. 318\)](#).

Chamar a API usando um cliente de API REST

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

Para chamar nossa API de proxy Amazon S3 usando o Postman

1. Implante ou reimplemente 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. Digite o ID de chave de acesso e a chave de acesso secreta do seu usuário do IAM nos campos de entrada AccessKey e SecretKey, respectivamente. Digite a região da AWS na qual a sua API é implantada na caixa de texto Região da AWS. Digite execute-api no campo de entrada Nome do serviço.

Você pode criar um par de chaves da guia Credenciais de segurança da sua conta de usuário do IAM no IAM Management Console.

4. Para adicionar um bucket denominado `apig-demo-5` à sua conta do Amazon S3 na região `{region}`:

Note

Certifique-se de que o nome do bucket seja globalmente exclusivo.

- 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 `apig-demo-5` 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/apig-demo-5/Readme.txt HTTP/1.1
Host: 9gn28ca086.execute-api.{region}.amazonaws.com
Content-Type: application/xml
X-Amz-Date: 20161015T062647Z
Authorization: AWS4-HMAC-SHA256 Credential=access-key-id/20161015/{region}/execute-
api/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=ccadb877bdb0d395ca38cc47e18a0d76bb5eaf17007d11e40bf6fb63d28c705b
```

```
Cache-Control: no-cache
Postman-Token: 6135d315-9cc4-8af8-1757-90871d00847e

Hello, World!
```

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 `apig-demo-5`, faça uma solicitação GET como a seguinte:

```
GET /S3/apig-demo-5/Readme.txt HTTP/1.1
Host: 9gn28ca086.execute-api.{region}.amazonaws.com
Content-Type: application/xml
X-Amz-Date: 20161015T063759Z
Authorization: AWS4-HMAC-SHA256 Credential=access-key-id/20161015/{region}/
execute-api/aws4_request, SignedHeaders=content-type;host;x-amz-date,
Signature=ba09b72b585acf0e578e6ad02555c00e24b420b59025bc7bb8d3f7aed1471339
Cache-Control: no-cache
Postman-Token: d60fc59-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 `apig-demo-5`, envie a seguinte solicitação:

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

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...603add12ee</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.

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

As seguintes definições do OpenAPI descrevem a API de amostra, referenciada neste tutorial, como um proxy Amazon S3.

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": {
          "type": "lambda",
          "uri": "arn:aws:lambda:{region}:{account}:my-s3-proxy",
          "httpMethod": "POST",
          "timeout": "30",
          "cacheControl": "no-store"
        }
      }
    }
  }
}
```

```
    "credentials": "arn:aws:iam::<replaceable>123456789012</replaceable>: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::<replaceable>123456789012</replaceable>: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::<replaceable>123456789012</replaceable>: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.header.x-amz-acl": "'authenticated-read'",
            "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::<replaceable>123456789012</replaceable>:role/
apigAwsProxyRole",
    "responses": {
        "4\b{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\b{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"
            }
        }
    }
}
```

```
"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::<replaceable>123456789012</replaceable>: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::<replaceable>123456789012</replaceable>: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::<replaceable>123456789012</replaceable>:role/apigAwsProxyRole",
        "responses": {
            "4\b\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.header.x-amz-acl": "'authenticated-read'",
        "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::<replaceable>123456789012</replaceable>:role/
apigAwsProxyRole",
        "type": "awsLambda"
    }
}
```

```
"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"
    }
}
}
```

TUTORIAL: Criar uma API REST como um proxy do Amazon Kinesis no API Gateway

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, você deve escolher uma região em que tanto os serviços do API Gateway quanto do Kinesis estejam disponíveis. Para conhecer a disponibilidade das regiões, consulte [Regiões e endpoints](#).

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

1. Listar os fluxos 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 esse método à ação `ListStreams` no Kinesis para listar os fluxos na conta do chamador.
- Expondo um método HTTP POST no recurso `/streams/{stream-name}` da API e integrando esse método à ação `CreateStream` no Kinesis para criar um fluxo nomeado na conta do chamador.
- Expondo um método HTTP GET no recurso `/streams/{stream-name}` da API e integrando esse método à ação `DescribeStream` no Kinesis para descrever um fluxo nomeado na conta do chamador.
- Expondo um método HTTP DELETE no recurso `/streams/{stream-name}` da API e integrando esse método à ação `DeleteStream` no Kinesis para excluir um fluxo na conta do chamador.
- Expondo um método HTTP PUT no recurso `/streams/{stream-name}/record` da API e integrando esse 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 esse 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.

Você pode aplicar as instruções apresentadas aqui a outras ações do Kinesis. Para a lista completa de ações do Kinesis, consulte o documento [Amazon Kinesis API Reference](#).

Em vez de usar o console do API Gateway para criar a API de amostra, você pode importar a API de amostra para o API Gateway usando a [API de importação](#) do API Gateway. Para obter informações sobre como usar o recurso Import API, consulte [Importar uma API REST no API Gateway \(p. 356\)](#).

Se você ainda não tiver uma conta da AWS, use o procedimento a seguir para criar uma.

Para cadastrar-se na AWS

1. Abra <https://aws.amazon.com/> e escolha Create an AWS Account.
2. Siga as instruções online.

Criar uma função e política IAM para a API para acessar o Kinesis

Para permitir que a API invoque ações do Kinesis, você deve ter as políticas IAM apropriadas anexadas a uma função IAM. Esta seção explica como verificar e criar, se necessário, as políticas e a função IAM necessárias.

Para habilitar o acesso somente leitura ao Kinesis, você pode usar a política `AmazonKinesisReadOnlyAccess`, que permite a invocação das ações `Get*`, `List*` e `Describe*` no Kinesis.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "kinesis:Get*",  
                "kinesis>List*",  
                "kinesis:Describe*"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

Essa política está disponível como uma política gerenciada provisionada pelo IAM, e seu ARN é `arn:aws:iam::aws:policy/AmazonKinesisReadOnlyAccess`.

Para habilitar ações de leitura/gravação no Kinesis, você pode usar a política `AmazonKinesisFullAccess`.

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

Essa política também está disponível como uma política gerenciada provisionada pelo IAM. Seu ARN é `arn:aws:iam::aws:policy/AmazonKinesisFullAccess`.

Depois de decidir qual política IAM deve ser usada, anexe-a a uma função IAM nova ou existente. Certifique-se de que o serviço de controle do API Gateway (`apigateway.amazonaws.com`) seja uma entidade confiável da função e tenha permissão para assumir a função de execução (`sts:AssumeRole`).

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

Se você criar a função de execução no console do IAM e escolher o tipo de função Amazon API Gateway, essa política de confiança será automaticamente anexada.

Observe o ARN da função de execução. Você precisará dela ao criar um método de API e configurar sua solicitação de integração.

Começar a criar uma API como um proxy do Kinesis

Use as seguintes etapas para criar a API no console do API Gateway.

Para criar uma API como um proxy de serviço da AWS para o Kinesis

1. No console API Gateway, escolha Create API.
2. Escolha Nova API.
3. Em API name (Nome da API), digite **KinesisProxy**. Deixe os valores padrão nos outros campos.
4. Se desejar, digite uma descrição em Description.
5. Escolha Create API (Criar API).

Após a criação da API, o console API Gateway exibe a página Resources, que contém apenas o recurso raiz (/) da API.

Listar fluxos do Kinesis

O Kinesis oferece suporte à 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. Em seguida, defina um método `GET` no recurso e integre o método com a ação `ListStreams` do Kinesis.

O procedimento a seguir descreve como listar fluxos do Kinesis usando o console do API Gateway.

Para listar fluxos do Kinesis usando o console do API Gateway

1. Selecione o recurso raiz da API. Em Actions, escolha Create Resource.

Em Resource Name, digite **Streams**, deixe Resource Path e outros campos com o padrão, e escolha Create Resource.

2. Escolha o recurso **/Streams**. Em Actions, escolha Create Method, escolha GET na lista e escolha o ícone de marca de seleção para concluir a criação do método.

Note

O verbo HTTP para um método invocado por um cliente pode ser diferente do verbo HTTP para uma integração exigida pelo back-end. Aqui, escolhemos **GET** porque a listagem de fluxos é intuitivamente uma operação READ.

3. No painel Setup, selecione AWS Service.

- a. Em AWS Region, escolha uma região (por exemplo, us-east-1).
- b. Em AWS Service, escolha Kinesis.
- c. Deixe AWS Subdomain em branco.
- d. Em Método HTTP, escolha POST.

Note

Aqui, escolhemos **POST** porque o Kinesis exige que a ação **ListStreams** seja chamada com ele.

- e. Em Action Type, escolha Use action name.
- f. Em Action, digite **ListStreams**.
- g. Em Execution role, digite o ARN para a sua função de execução.
- h. Deixe o padrão de Passthrough para Content Handling.
- i. Escolha Save para concluir a instalação inicial do método.

/streams - GET - Setup



Choose the integration point for your new method.

Integration type Lambda Function (i)

HTTP (i)

Mock (i)

AWS Service (i)

AWS Region (i)

AWS Service (i)

AWS Subdomain

HTTP method (i)

Action Type Use action name

Use path override

Action

Execution role (i)

Content Handling (i)

Save

4. Ainda no painel Integration Request, expanda a seção HTTP Headers:

- a. Escolha Add header (Adicionar cabeçalho).
- b. Na coluna Name, digite Content-Type.
- c. Na coluna Mapped from, digite 'application/x-amz-json-1.1'.
- d. Escolha o ícone de marca de seleção para salvar a configuração.

Usamos um mapeamento de parâmetros de solicitação para definir o cabeçalho Content-Type para o valor estático do 'application/x-amz-json-1.1' para informar o Kinesis que a entrada é de uma versão específica do JSON.

5. Expand a seção Body Mapping Templates:

- a. Escolha Add mapping template (Adicionar modelo de mapeamento).
- b. Em Content-Type, digite application/json.
- c. Escolha o ícone de marca de seleção para salvar a configuração Content-Type. Escolha Yes, secure this integration em Change passthrough behavior.

- d. Digite {} no editor de modelo.
- e. Escolha o botão Save para salvar o modelo de mapeamento.

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.

▼ HTTP Headers

Name	Mapped from ⓘ	Caching
Content-Type	'application/x-amz-json-1.1'	<input type="checkbox"/>  

+ Add header

▼ Body Mapping Templates

Request body passthrough When there are no templates defined (recommended) ⓘ When no template matches the request Content-Type header ⓘ Never ⓘ

Content-Type
application/json 

+ Add mapping template

application/json

Generate template:

1 [{}]



Cancel **Save**

6. Testar o método GET no recurso Streams para chamar a ação `ListStreams` no Kinesis:

No console do API Gateway, selecione a entrada `/streams/GET` do painel Resources, escolha a opção de invocação `Test` e, então, escolha `Test`.

Se você já tiver criado dois fluxos chamados "myStream" e "yourStream" no Kinesis, o teste bem-sucedido retornará uma resposta 200 OK contendo a seguinte carga:

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

Criar, descrever e excluir um fluxo no Kinesis

Criar, descrever e excluir um fluxo no Kinesis envolve 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  
  
{  
    "ExclusiveStartShardId": "string",  
    "Limit": number,  
    "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 fluxo 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.

Para configurar e testar o método GET em um recurso de fluxo

1. Crie um recurso filho com a variável de caminho {stream-name} sob o recurso /streams criado anteriormente.

New Child Resource

Use this page to create a new child resource for your resource.

Configure as proxy resource



Stream-name

Resource Path*

/streams/{stream-name}

You can add path parameters using brackets. For example, the resource path `{username}` represents a path parameter called 'username'. Configuring `/streams/{proxy+}` as a proxy resource catches all requests to its sub-resources. For example, it works for a GET request to `/streams/foo`. To handle requests to `/streams`, add a new ANY method on the `/streams` resource.

Enable API Gateway CORS



* Required

Cancel

Create Resource

2. Adicione os verbos HTTP POST, GET e DELETE a este recurso.

Depois que os métodos são criados no recurso, a estrutura da API é semelhante à seguinte:

Resources Actions ▾

▼  /

 ▼  /streams

 GET

 ▼  /{stream-name}

 DELETE

 POST

 GET

3. Configure o método GET /streams/{stream-name} para chamar a ação POST /?Action=DescribeStream no Kinesis, conforme mostrado a seguir.

/streams/{stream-name} - GET - Setup



Choose the integration point for your new method.

Integration type Lambda Function [?](#)
 HTTP [?](#)
 Mock [?](#)
 AWS Service [?](#)

AWS Region

AWS Service

AWS Subdomain

HTTP method

Action Type Use action name
 Use path override

Action

Execution role

Content Handling

Save

4. Adicione o seguinte mapeamento do cabeçalho Content-Type à solicitação de integração:

```
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.

5. 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.

6. Teste o método GET /stream/{stream-name} para invocar a ação DescribeStream no Kinesis:

No console do API Gateway, selecione `/streams/{stream-name}/GET` no painel Resources, escolha Test para iniciar o teste, digite o nome de um fluxo existente do Kinesis no campo Path para `stream-name` e depois escolha Test. 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. Configure o método POST `/streams/{stream-name}` para chamar a ação POST `/?` Action=CreateStream no Kinesis. A tarefa segue o mesmo procedimento para configurar o método GET `/streams/{stream-name}` desde que você substitua a ação `DescribeStream` por `CreateStream`.
2. Adicione o seguinte mapeamento do cabeçalho Content-Type à solicitação de integração:

```
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.

3. 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.

4. Teste o método POST /streams/{stream-name} para criar um fluxo nomeado no Kinesis:

No console do API Gateway, selecione /streams/{stream-name}/POST no painel Resources, escolha Test para iniciar o teste, digite o nome de um fluxo existente do Kinesis em Path para stream-name e depois escolha Test. Se o teste for bem-sucedido, uma resposta 200 OK será retornado sem dados.

Depois de implantar a API, você também poderá 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. Configure o método DELETE /streams/{stream-name} para integrar-se com a ação POST /?Action=DeleteStream no Kinesis. A tarefa segue o mesmo procedimento para configurar o método GET /streams/{stream-name} desde que você substitua a ação DescribeStream por DeleteStream.
2. Adicione o seguinte mapeamento do cabeçalho Content-Type à solicitação de integração:

```
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.

3. 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.

4. Teste o método `DELETE` para excluir um fluxo nomeado no Kinesis:

No console do API Gateway, selecione o nó de método `/streams/{stream-name}/DELETE` no painel Resources, escolha Test para iniciar o teste, digite o nome de um fluxo existente do Kinesis no campo Path para `stream-name` e depois escolha Test. Se o teste for bem-sucedido, uma resposta 200 OK será retornado sem dados.

Depois de implantar a API, você também poderá fazer a seguinte solicitação de API REST 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 fluxo no Kinesis

Depois de criar um fluxo no Kinesis, você poderá adicionar registros de dados ao fluxo e ler os dados desse fluxo. 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 fluxo de origem do qual estamos obtendo registros é especificado no valor ShardIterator necessário, conforme indicado na seguinte ação do Kinesis para obter um iterador de estilhaços:

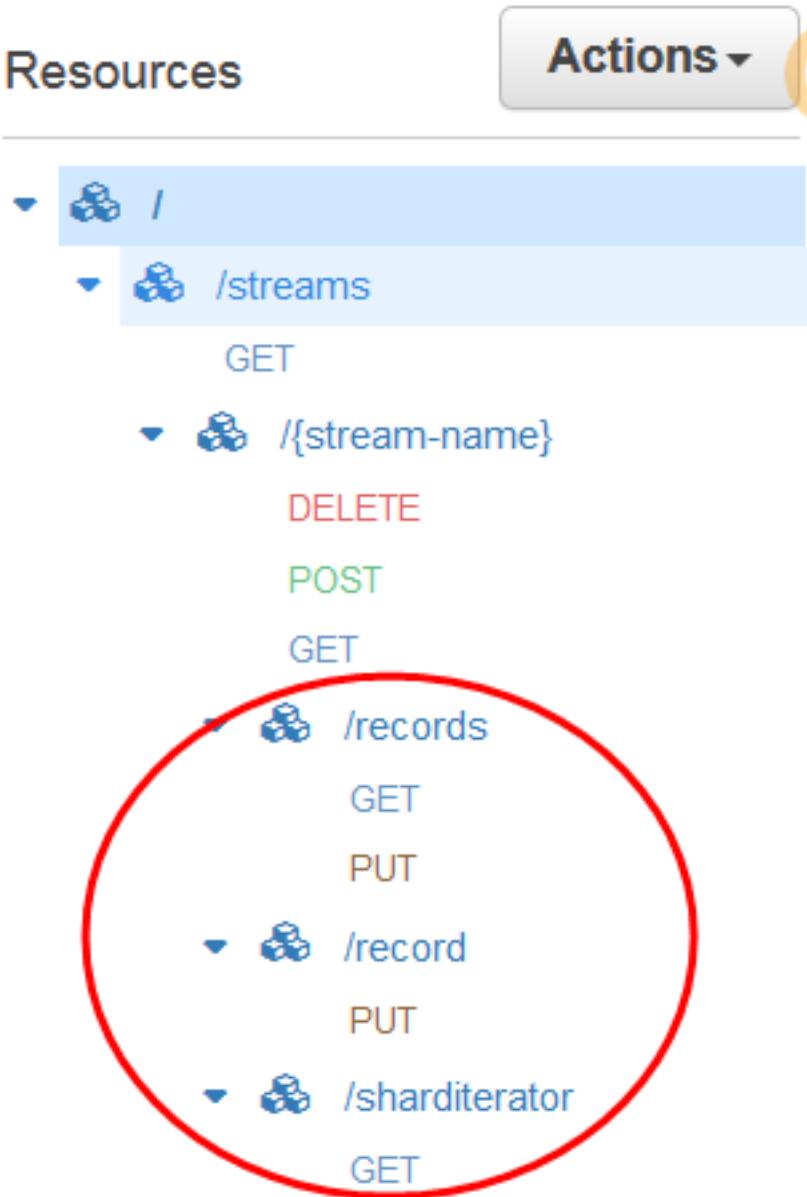
```
POST /?Action=GetShardIterator HTTP/1.1
Host: kinesis.region.domain
Authorization: AWS4-HMAC-SHA256 Credential=..., ...
...
Content-Type: application/x-amz-json-1.1
Content-Length: PayloadSizeBytes

{
    "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).

A figura a seguir mostra a estrutura de API dos recursos após a criação dos métodos:



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.

Para configurar e testar o método **PUT /streams/{stream-name}/record** para invocar **PutRecord** no Kinesis:

1. Configure o método PUT, conforme mostrado a seguir:

[Method Execution](#)

/streams/{stream-name}/record - PUT - Integration Request

Provide information about the target backend that this method will call and whether the incoming request data should be modified.

Integration type Lambda Function [i](#)
 HTTP [i](#)
 Mock [i](#)
 AWS Service [i](#)

AWS Region us-east-1 [e](#)

AWS Service Kinesis [e](#)

AWS Subdomain [e](#)

HTTP method POST [e](#)

Action PutRecord [e](#)

Execution role arn:aws:iam::█████████████████████:role/apigAwsProxyRole [e](#)

Credentials cache Do not add caller credentials to cache key [e](#)

Content Handling Passthrough [e](#) [i](#)

2. Adicione o seguinte mapeamento do parâmetro de solicitação para definir o cabeçalho Content-Type como uma versão do JSON compatível com a AWS na solicitação de integração:

```
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.

3. 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.

4. 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-000000000004"  
}
```

Para configurar e testar o método `PUT /streams/{stream-name}/records` para invocar `PutRecords` no Kinesis:

1. Configure o método `PUT /streams/{stream-name}/records`, conforme mostrado a seguir:

◀ Method Execution
[/streams/{stream-name}/records - PUT - Integration Request](#) 

Provide information about the target backend that this method will call and whether the incoming request data should be modified.

Integration type Lambda Function 
 HTTP 
 Mock 
 AWS Service 

AWS Region us-east-1 

AWS Service Kinesis 

AWS Subdomain 

HTTP method POST 

Action PutRecords 

Execution role arn:aws:iam::7...:role/apigAwsProxyRole 

Credentials cache Do not add caller credentials to cache key 

Content Handling Passthrough 

2. Adicione o seguinte mapeamento do parâmetro de solicitação para definir o cabeçalho Content-Type como uma versão do JSON compatível com a AWS na solicitação de integração:

```
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.

3. Adicione o seguinte modelo de mapeamento de corpo para mapear 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 back-end 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.

4. 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"  
        }  
    ]  
}
```

```
    ]  
}
```

Para 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. Configure a integração do método **GET /streams/{stream-name}/sharditerator**, conforme mostrado a seguir:

[Method Execution](#)

/streams/{stream-name}/sharditerator - GET - Integration ...

Provide information about the target backend that this method will call and whether the incoming request data should be modified.

Integration type Lambda Function [i](#)
 HTTP [i](#)
 Mock [i](#)
 AWS Service [i](#)

AWS Region us-east-1 [e](#)

AWS Service Kinesis [e](#)

AWS Subdomain [e](#)

HTTP method POST [e](#)

Action GetShardIterator [e](#)

Execution role arn:aws:iam::7...7:role/apigAwsProxyRole [e](#)

Credentials cache Do not add caller credentials to cache key [e](#)

Content Handling Passthrough [e](#) [i](#)

2. 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 a seguir:

◀ Method Execution
[/streams/{stream-name}/sharditerator - GET - Method Req...](#) 

Provide information about this method's authorization settings and the parameters it can receive.

Settings

Authorization NONE  

Request Validator NONE  

API Key Required false 

▶ Request Paths

▼ URL Query String Parameters

Name	Required	Caching	
shard-id	<input type="checkbox"/>	<input type="checkbox"/>	 

 [Add query string](#)

▶ HTTP Request Headers

▶ Request Body 

▶ SDK Settings

No modelo de mapeamento de corpo a seguir, definimos o valor do parâmetro de consulta `shard-id` como o valor da propriedade `ShardId` da carga do JSON como a entrada para a ação `GetShardIterator` no Kinesis.

3. Configure o modelo de mapeamento de corpo para gerar a entrada necessária (`ShardId` e `StreamName`) para a ação `GetShardIterator` nos 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')"  
}
```

4. Usando a opção Test no console API Gateway, insira um nome de fluxo existente como o valor da variável `stream-name` Path, defina `shard-id` Query string como um valor `ShardId` existente (por exemplo, `shard-000000000004`), e escolha Test.

A carga da resposta bem-sucedida é semelhante à saída a seguir:

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

Anote o valor de **ShardIterator**. Você precisa dela para obter registros de um fluxo.

Para configurar e testar o método **GET /streams/{stream-name}/records** para invocar a ação **GetRecords** no Kinesis

1. Configure o método **GET /streams/{stream-name}/records**, conforme mostrado a seguir:

[Method Execution](#)

/streams/{stream-name}/records - GET - Integration Request

Provide information about the target backend that this method will call and whether the incoming request data should be modified.

Integration type Lambda Function [i](#)
 HTTP [i](#)
 Mock [i](#)
 AWS Service [i](#)

AWS Region us-east-1 [e](#)

AWS Service Kinesis [e](#)

AWS Subdomain [e](#)

HTTP method POST [e](#)

Action GetRecords [e](#)

Execution role arn:aws:iam::777777777777:role/apigAwsProxyRole [e](#)

Credentials cache Do not add caller credentials to cache key [e](#)

Content Handling Passthrough [e](#) [i](#)

2. 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, conforme mostrado a seguir:

[Method Execution](#) /streams/{stream-name}/records - GET - Method Request 

Provide information about this method's authorization settings and the parameters it can receive.

Settings

Authorization NONE  

Request Validator NONE  

API Key Required false 

▶ Request Paths

▶ URL Query String Parameters

▼ HTTP Request Headers

Name	Required	Caching	
Shard-Iterator	<input type="checkbox"/>	<input type="checkbox"/>	 

 Add header

▶ Request Body 

▶ SDK Settings

3. Configure o seguinte modelo de mapeamento para mapear o valor do parâmetro de cabeçalho **Shard-Iterator** para o valor da propriedade **ShardIterator** da carga do JSON para a ação **GetRecords** no Kinesis.

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

4. Usando a opção Test no console API Gateway, digite um nome de fluxo existente como o valor da variável **stream-name** Path, defina **Shard-Iterator** Header como o valor **ShardIterator** obtido na execução do teste do método GET `/streams/{stream-name}/sharditerator` (acima), e escolha Test.

A carga da resposta bem-sucedida é semelhante à saída a seguir:

```
{  
    "MillisBehindLatest": 0,  
    "NextShardIterator": "AAAAAAAAAAF...",  
    "Records": [ ... ]  
}
```

Definições do OpenAPI de uma API de exemplo como um proxy do Kinesis

Veja a seguir as definições do OpenAPI para a API de amostra como um proxy Kinesis usado neste tutorial.

OpenAPI 3.0

```
{
  "openapi": "3.0.0",
  "info": {
    "version": "2016-03-31T18:25:32Z",
    "title": "KinesisProxy"
  },
  "paths": {
    "/streams": {
      "get": {
        "responses": {
          "200": {
            "description": "200 response",
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Empty"
                }
              }
            }
          }
        }
      },
      "x-amazon-apigateway-integration": {
        "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
        "responses": {
          "default": {
            "statusCode": "200"
          }
        },
        "requestTemplates": {
          "application/json": "{\n}"
        },
        "uri": "arn:aws:apigateway:us-east-1:kinesis:action/ListStreams",
        "httpMethod": "POST",
        "requestParameters": {
          "integration.request.header.Content-Type": "'application/x-amz-json-1.1'"
        },
        "type": "aws"
      }
    },
    "/streams/{stream-name)": {
      "get": {
        "parameters": [
          {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "content": {
              "application/json": {
                "schema": {
                  "$ref": "#/components/schemas/Empty"
                }
              }
            }
          }
        }
      }
    }
  }
}
```

```
"200": {
    "description": "200 response",
    "content": {
        "application/json": {
            "schema": {
                "$ref": "#/components/schemas/Empty"
            }
        }
    }
},
"x-amazon-apigateway-integration": {
    "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
    "responses": {
        "default": {
            "statusCode": "200"
        }
    },
    "requestTemplates": {
        "application/json": "{\n          \"StreamName\": \"$input.params('stream-\nname')\"\n        },\n        \"uri\": \"arn:aws:apigateway:us-east-1:kinesis:action/DescribeStream\", \n        \"httpMethod\": \"POST\", \n        \"type\": \"aws\"\n    }
},
"post": {
    "parameters": [
        {
            "name": "stream-name",
            "in": "path",
            "required": true,
            "schema": {
                "type": "string"
            }
        }
    ],
    "responses": {
        "200": {
            "description": "200 response",
            "content": {
                "application/json": {
                    "schema": {
                        "$ref": "#/components/schemas/Empty"
                    }
                }
            }
        }
    }
},
"x-amazon-apigateway-integration": {
    "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
    "responses": {
        "default": {
            "statusCode": "200"
        }
    },
    "requestTemplates": {
        "application/json": "{\n          \"ShardCount\": 5,\n          \"StreamName\": \"$input.params('stream-name')\"\n        },\n        \"uri\": \"arn:aws:apigateway:us-east-1:kinesis:action/CreateStream\", \n        \"httpMethod\": \"POST\", \n        \"requestParameters\": {\n            \"integration.request.header.Content-Type\": \"application/x-amz-\njson-1.1\""
    }
}
```

```
        },
        "type": "aws"
    }
},
"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"
                    }
                }
            }
        },
        "500": {
            "description": "500 response",
            "headers": {
                "Content-Type": {
                    "schema": {
                        "type": "string"
                    }
                }
            }
        }
    }
},
"x-amazon-apigateway-integration": {
    "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
    "responses": {
        "4\d{2}": {
            "statusCode": "400",
            "responseParameters": {
                "method.response.header.Content-Type": "application/json"
            }
        },
        "default": {
            "statusCode": "200",
            "responseParameters": {
                "method.response.header.Content-Type": "application/json"
            }
        }
    }
}
```

```
        "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"
        }
    }
},
"requestTemplates": {
    "application/json": "{\n        \"StreamName\": \"$input.params('stream-name')\"\n    },\n    \"uri\": \"arn:aws:apigateway:us-east-1:kinesis:action/DeleteStream\", \n    \"httpMethod\": \"POST\", \n    \"requestParameters\": {\n        \"integration.request.header.Content-Type\": \"application/x-amz-json-1.1\"\n    },\n    \"type\": \"aws\"\n}
},
"/streams/{stream-name}/record": {
    "put": {
        "parameters": [
            {
                "name": "stream-name",
                "in": "path",
                "required": true,
                "schema": {
                    "type": "string"
                }
            }
        ],
        "responses": {
            "200": {
                "description": "200 response",
                "content": {
                    "application/json": {
                        "schema": {
                            "$ref": "#/components/schemas/Empty"
                        }
                    }
                }
            }
        }
    },
    "x-amazon-apigateway-integration": {
        "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
        "responses": {
            "default": {
                "statusCode": "200"
            }
        },
        "requestTemplates": {
            "application/json": "{\n        \"StreamName\": \"$input.params('stream-name')\", \n        \"Records\": [\n            #foreach($elem in $input.path('$records'))\n            {\n                \"Data\": \"$util.base64Encode($elem.data)\",\n                \"PartitionKey\": \"$elem.partition-key\"\n            }#if($foreach.hasNext),#\n            #end\n        ]\n    },\n    \"uri\": \"arn:aws:apigateway:us-east-1:kinesis:action/PutRecord\", \n    \"httpMethod\": \"POST\", \n    \"method.response.header.Content-Type\": \"application/json\"\n}\n"
    }
}
```

```
"requestParameters": {
    "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
},
    "type": "aws"
}
},
"/streams/{stream-name}/records": {
    "get": {
        "parameters": [
            {
                "name": "stream-name",
                "in": "path",
                "required": true,
                "schema": {
                    "type": "string"
                }
            },
            {
                "name": "Shard-Iterator",
                "in": "header",
                "required": false,
                "schema": {
                    "type": "string"
                }
            }
        ],
        "responses": {
            "200": {
                "description": "200 response",
                "content": {
                    "application/json": {
                        "schema": {
                            "$ref": "#/components/schemas/Empty"
                        }
                    }
                }
            }
        }
    },
    "x-amazon-apigateway-integration": {
        "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
        "responses": {
            "default": {
                "statusCode": "200"
            }
        },
        "requestTemplates": {
            "application/json": "{\n    \"ShardIterator\": \"$input.params('Shard-Iterator')\"\n}"
        },
        "uri": "arn:aws:apigateway:us-east-1:kinesis:action/GetRecords",
        "httpMethod": "POST",
        "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-
json-1.1'"
},
        "type": "aws"
}
},
    "put": {
        "parameters": [
            {
                "name": "Content-Type",
                "in": "header",
                "required": false,
```

```
        "schema": {
            "type": "string"
        }
    },
    {
        "name": "stream-name",
        "in": "path",
        "required": true,
        "schema": {
            "type": "string"
        }
    }
],
"responses": {
    "200": {
        "description": "200 response",
        "content": {
            "application/json": {
                "schema": {
                    "$ref": "#/components/schemas/Empty"
                }
            }
        }
    }
},
"x-amazon-apigateway-integration": {
    "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
    "responses": {
        "default": {
            "statusCode": "200"
        }
    },
    "requestTemplates": {
        "application/json": "{\n            \"StreamName\": \"$input.params('stream-name')\", \n            \"Records\": [\n                {\n                    \"Data\": \"$util.base64Encode($elem.data)\",\n                    \"PartitionKey\": \"$elem.partition-key\"\n                }#if($foreach.hasNext),#end\n            ]\n        },\n        \"application/x-amz-json-1.1\": \"{\\n            \"StreamName\": \"$input.params('stream-name')\",\\n            \"records\": [\n                {\\n                    \"Data\": \"$elem.data\",\\n                    \"PartitionKey\": \"$elem.partition-key\"\n                }#if($foreach.hasNext),#end\n            ]\\n        }\""
    },
    "uri": "arn:aws:apigateway:us-east-1:kinesis:action/PutRecords",
    "httpMethod": "POST",
    "requestParameters": {
        "integration.request.header.Content-Type": "'application/x-amz-json-1.1'"
    },
    "type": "aws"
},
"requestBody": {
    "content": {
        "application/json": {
            "schema": {
                "$ref": "#/components/schemas/PutRecordsMethodRequestPayload"
            }
        },
        "application/x-amz-json-1.1": {
            "schema": {
                "$ref": "#/components/schemas/PutRecordsMethodRequestPayload"
            }
        }
    },
    "required": true
}
}
```

```
        },
        "/streams/{stream-name}/sharditerator": {
            "get": {
                "parameters": [
                    {
                        "name": "stream-name",
                        "in": "path",
                        "required": true,
                        "schema": {
                            "type": "string"
                        }
                    },
                    {
                        "name": "shard-id",
                        "in": "query",
                        "required": false,
                        "schema": {
                            "type": "string"
                        }
                    }
                ],
                "responses": {
                    "200": {
                        "description": "200 response",
                        "content": {
                            "application/json": {
                                "schema": {
                                    "$ref": "#/components/schemas/Empty"
                                }
                            }
                        }
                    }
                }
            },
            "x-amazon-apigateway-integration": {
                "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
                "responses": {
                    "default": {
                        "statusCode": "200"
                    }
                },
                "requestTemplates": {
                    "application/json": "{\n          \"ShardId\": \"$input.params('shard-\n          id')\", \n          \"ShardIteratorType\": \"TRIM_HORIZON\", \n          \"StreamName\":\n          \"$input.params('stream-name')\"\n        },\n        \"uri\": \"arn:aws:apigateway:us-east-1:kinesis:action/GetShardIterator\", \n        \"httpMethod\": \"POST\", \n        \"requestParameters\": {\n          \"integration.request.header.Content-Type\": \"application/x-amz-\n          json-1.1\"\n        },\n        \"type\": \"aws\"\n      }\n    }\n  },\n  \"servers\": [\n    {\n      \"url\": \"https://wd4zclrobb.execute-api.us-east-1.amazonaws.com/{basePath}\",\n      \"variables\": {\n        \"basePath\": {\n          \"default\": \"/test\"\n        }\n      }\n    }\n  ],
```

```
"components": {
  "schemas": {
    "PutRecordsMethodRequestPayload": {
      "type": "object",
      "properties": {
        "records": {
          "type": "array",
          "items": {
            "type": "object",
            "properties": {
              "data": {
                "type": "string"
              },
              "partition-key": {
                "type": "string"
              }
            }
          }
        }
      }
    },
    "Empty": {
      "type": "object"
    }
  }
}
```

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "version": "2016-03-31T18:25:32Z",
    "title": "KinesisProxy"
  },
  "host": "wd4zclrobb.execute-api.us-east-1.amazonaws.com",
  "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": {
        "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
        "responses": {
          "default": {
            "statusCode": "200"
          }
        }
      }
    }
  }
}
```

```
    "requestTemplates": {
        "application/json": "{\n    \"uri\": \"arn:aws:apigateway:us-east-1:kinesis:action/ListStreams\", \"httpMethod\": \"POST\", \"requestParameters\": { \"integration.request.header.Content-Type\": \"application/x-amz-json-1.1\" }, \"type\": \"aws\" }\n    }\n},
"/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": {
                    "$ref": "#/definitions/Empty"
                }
            }
        },
        "x-amazon-apigateway-integration": {
            "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
            "responses": {
                "default": {
                    "statusCode": "200"
                }
            },
            "requestTemplates": {
                "application/json": "{$input.params('stream-name')\"\n    \"uri\": \"arn:aws:apigateway:us-east-1:kinesis:action/DescribeStream\", \"httpMethod\": \"POST\", \"type\": \"aws\" }\n    }\n",
                "post": {
                    "consumes": [
                        "application/json"
                    ],
                    "produces": [
                        "application/json"
                    ],
                    "parameters": [
                        {
                            "name": "stream-name",
                            "in": "path",
                            "required": true,
                            "type": "string"
                        }
                    ]
                }
            }
        }
    }
}
```

```
        ],
        "responses": {
            "200": {
                "description": "200 response",
                "schema": {
                    "$ref": "#/definitions/Empty"
                }
            }
        },
        "x-amazon-apigateway-integration": {
            "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
            "responses": {
                "default": {
                    "statusCode": "200"
                }
            },
            "requestTemplates": {
                "application/json": "{\n        \"ShardCount\": 5,\n        \"StreamName\": \"$input.params('stream-name')\"\n    },\n    \"uri\": \"arn:aws:apigateway:us-east-1:kinesis:action/CreateStream\","
                "httpMethod": "POST",
                "requestParameters": {
                    "integration.request.header.Content-Type": "'application/x-amz-json-1.1'"
                },
                "type": "aws"
            },
            "delete": {
                "consumes": [
                    "application/json"
                ],
                "produces": [
                    "application/json"
                ],
                "parameters": [
                    {
                        "name": "stream-name",
                        "in": "path",
                        "required": true,
                        "type": "string"
                    }
                ],
                "responses": {
                    "200": {
                        "description": "200 response",
                        "schema": {
                            "$ref": "#/definitions/Empty"
                        },
                        "headers": {
                            "Content-Type": {
                                "type": "string"
                            }
                        }
                    },
                    "400": {
                        "description": "400 response",
                        "headers": {
                            "Content-Type": {
                                "type": "string"
                            }
                        }
                    },
                    "500": {
                        "description": "500 response",
                        "headers": {
                            "Content-Type": {
                                "type": "string"
                            }
                        }
                    }
                }
            }
        }
    }
}
```

```
        "Content-Type": {
            "type": "string"
        }
    }
},
"x-amazon-apigateway-integration": {
    "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
    "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"
        }
    }
},
"requestTemplates": {
    "application/json": "{\n      \"StreamName\": \"$input.params('stream-name')\"\n    },\n    \"uri\": \"arn:aws:apigateway:us-east-1:kinesis:action/DeleteStream\", \n    \"httpMethod\": \"POST\", \n    \"requestParameters\": { \n      \"integration.request.header.Content-Type\": 'application/x-amz-json-1.1' \n    }, \n    \"type\": \"aws\" \n  }\n},\n"/streams/{stream-name}/record": {
    "put": {
        "consumes": [
            "application/json"
        ],
        "produces": [
            "application/json"
        ],
        "parameters": [
            {
                "name": "stream-name",
                "in": "path",
                "required": true,
                "type": "string"
            }
        ],
        "responses": {
            "200": {
                "description": "200 response",
                "schema": {
                    "$ref": "#/definitions/Empty"
                }
            }
        }
    }
}
```

```
        },
    },
    "x-amazon-apigateway-integration": {
        "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
        "responses": {
            "default": {
                "statusCode": "200"
            }
        },
        "requestTemplates": {
            "application/json": "{\n              \"StreamName\": \"$input.params('stream-name')\", \n              \"Records\": [\n                #foreach($elem in $input.path('$records'))\n                  {\n                    \"Data\": \"$util.base64Encode($elem.data)\",\n                    \"PartitionKey\": \"$elem.partition-key\"\n                  }#if($foreach.hasNext),#end\n                #end\n              ]\n            },\n            \"uri\": \"arn:aws:apigateway:us-east-1:kinesis:action/PutRecord\", \n            \"httpMethod\": \"POST\", \n            \"requestParameters\": {\n                \"integration.request.header.Content-Type\": \"application/x-amz-json-1.1\" \n            },\n            \"type\": \"aws\"\n          }\n        }\n    },
    "/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": {
                        "$ref": "#/definitions/Empty"
                    }
                }
            }
        },
        "x-amazon-apigateway-integration": {
            "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
            "responses": {
                "default": {
                    "statusCode": "200"
                }
            },
            "requestTemplates": {
                "application/json": "{\n                  \"ShardIterator\": \"$input.params('Shard-Iterator')\"\n                },\n                \"uri\": \"arn:aws:apigateway:us-east-1:kinesis:action/GetRecords\", \n              }\n            }\n        }\n    }
}
```

```
"httpMethod": "POST",
"requestParameters": {
    "integration.request.header.Content-Type": "'application/x-amz-json-1.1'"
},
"type": "aws"
},
"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": {
                "$ref": "#/definitions/PutRecordsMethodRequestPayload"
            }
        }
    ],
    "responses": {
        "200": {
            "description": "200 response",
            "schema": {
                "$ref": "#/definitions/Empty"
            }
        }
    }
},
"x-amazon-apigateway-integration": {
    "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
    "responses": {
        "default": {
            "statusCode": "200"
        }
    },
    "requestTemplates": {
        "application/json": "{\n    \"StreamName\": \"$input.params('stream-name')\", \n    \"Records\": [\n        #foreach($elem in $input.path('$..records'))\n        {\n            \"Data\": \"$util.base64Encode($elem.data)\"\n        }\n    ]\n},\n    \"application/x-amz-json-1.1\": \"$set($inputRoot = $input.path('$'))\n$inputRoot\n    \"StreamName\": \"$input.params('stream-name')\", \n    \"records\" : [\n        #foreach($elem in $inputRoot.records)\n            {\n                \"Data\" : \"$elem.data\", \n                \"PartitionKey\" : \"$elem.partition-key\"\n            }\n    ]\n}\n",
        "uri": "arn:aws:apigateway:us-east-1:kinesis:action/PutRecords",
        "httpMethod": "POST",
    }
}
```

```
        "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-json-1.1'"
        },
        "type": "aws"
    }
},
"/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": {
                    "$ref": "#/definitions/Empty"
                }
            }
        }
    },
    "x-amazon-apigateway-integration": {
        "credentials": "arn:aws:iam::123456789012:role/apigAwsProxyRole",
        "responses": {
            "default": {
                "statusCode": "200"
            }
        },
        "requestTemplates": {
            "application/json": "{$input.params('shard-id')}\n      \"ShardIteratorType\": \"TRIM_HORIZON\"\n      \"StreamName\": \"$input.params('stream-name')\"\n    "
        },
        "uri": "arn:aws:apigateway:us-east-1:kinesis:action/GetShardIterator",
        "httpMethod": "POST",
        "requestParameters": {
            "integration.request.header.Content-Type": "'application/x-amz-json-1.1'"
        },
        "type": "aws"
    }
},
"definitions": {
    "PutRecordsMethodRequestPayload": {
        "type": "object",
        "properties": {
            "records": {
                "type": "array",
                "items": {

```

```
"type": "object",
"properties": {
    "data": {
        "type": "string"
    },
    "partition-key": {
        "type": "string"
    }
}
},
"Empty": {
    "type": "object"
}
}
}
```

Criar, implantar e invocar uma API REST no Amazon API Gateway

Tópicos

- [Criação de uma API REST no Amazon API Gateway \(p. 182\)](#)
- [Controlar e gerenciar acesso a uma API REST no API Gateway \(p. 361\)](#)
- [Documentação de uma API REST no API Gateway \(p. 466\)](#)
- [Atualizar e manter uma API REST no Amazon API Gateway \(p. 510\)](#)
- [Implantando uma API REST no Amazon API Gateway \(p. 516\)](#)
- [Como invocar uma API REST no Amazon API Gateway \(p. 563\)](#)
- [Rastreamento, registro em logs e monitoramento de uma API do API Gateway \(p. 586\)](#)
- [Extensões do API Gateway para OpenAPI \(p. 606\)](#)

Criação de uma API REST no Amazon 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 pode usar um recurso [RestApi](#) para representar uma API que pode conter uma coleção de entidades [Resource](#). Cada entidade [Resource](#) pode, por sua vez, ter um ou mais recursos [Method](#). Expresso nos parâmetros de solicitação e corpo, um [Method](#) define a interface de programação de aplicativo para o cliente acessar o [Resource](#) exposto e representa uma solicitação recebida pelo cliente. Em seguida, você cria um recurso [Integration](#) para integrar o [Method](#) com um endpoint de back-end, também conhecido como o endpoint de integração, encaminhando a solicitação recebida para um URI de endpoint de integração especificado. Se necessário, você pode transformar os parâmetros de solicitação ou corpo para atender aos requisitos de back-end. Para as respostas, é possível criar um recurso [MethodResponse](#) para representar uma resposta de solicitação recebida pelo cliente e você pode criar um recurso [IntegrationResponse](#) para representar a resposta de solicitação retornada pelo back-end. Você pode configurar a resposta de integração para transformar os dados de resposta de back-end antes de retornar os dados para o cliente ou transmitir a resposta do back-end da forma em que se encontra para o cliente.

Para ajudar seus clientes a compreender sua API, você também pode fornecer sua documentação como parte da criação da API ou depois de criá-la. 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 \(p. 378\)](#), um [autorizador do Lambda \(p. 396\)](#) ou um [grupo de usuários do Amazon Cognito \(p. 414\)](#). Para medir o uso da sua API, configure [planos de uso \(p. 450\)](#) para limitar as solicitações de API. Você pode ativá-los ao criar ou atualizar a API.

É possível executar essas e outras tarefas usando o console do API Gateway, a API REST do API Gateway, a AWS CLI ou um dos AWS SDKs. Abordaremos como executar essas tarefas a seguir.

Tópicos

- [Escolher um tipo de endpoint para configurar uma API do API Gateway \(p. 183\)](#)
- [Inicializar a configuração da API REST no API Gateway \(p. 184\)](#)
- [Configurar métodos de API REST no API Gateway \(p. 204\)](#)

- [Configurar integrações da API REST no API Gateway \(p. 218\)](#)
- [Configurar respostas do gateway para personalizar respostas de erro \(p. 263\)](#)
- [Configurar mapeamentos de dados de solicitações e respostas do API Gateway \(p. 270\)](#)
- [Oferecer suporte a cargas úteis binárias no API Gateway \(p. 316\)](#)
- [Habilitar compactação de carga para uma API \(p. 339\)](#)
- [Ativar a validação de solicitação no API Gateway \(p. 343\)](#)
- [Importar uma API REST no API Gateway \(p. 356\)](#)

Escolher um tipo de endpoint para configurar uma API do API Gateway

Um tipo de [endpoint da API \(p. 6\)](#) refere-se ao nome do host da API. O tipo de endpoint da API pode ser otimizado para fronteiras, regional ou privado, dependendo de onde a maior parte do seu tráfego de API se origina.

Endpoint de API otimizado para fronteiras

Um [endpoint de API otimizado para fronteiras \(p. 7\)](#) é melhor para clientes distribuídos geograficamente. As solicitações de API são direcionadas para o ponto de presença (POP) mais próximo do CloudFront. 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 da API regional \(p. 8\)](#) 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 carga da conexão. Para obter mais informações, consulte [the section called “Configurar uma API regional” \(p. 197\)](#).

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 personalizado regional para a API WebSocket ou REST” \(p. 692\)](#) e [the section called “Como criar um nome de domínio personalizado otimizado para fronteiras” \(p. 685\)](#).

Os endpoints de API regionais transmitem todos os nomes de cabeçalho no estado em que se encontram.

Endpoints privados de API

Um [endpoint privado da API \(p. 8\)](#) 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 “Criar uma API privada” \(p. 200\)](#).

Os endpoints privados de API transmitem todos os nomes de cabeçalho no estado em que se encontram.

Inicializar a configuração da API REST no API Gateway

Para este exemplo, usamos uma API [PetStore](#) simplificada, com a integração de HTTP, que expõe os métodos GET /pets e GET /pets/{petId}. Os métodos são integrados com os dois endpoints HTTP, respectivamente, de `http://petstore-demo-endpoint.execute-api.com/petstore/pets` e `http://petstore-demo-endpoint.execute-api.com/petstore/pets/{petId}`. A API lida com respostas 200 OK. O exemplo se concentra nas tarefas de programação essenciais para criar uma API no API Gateway, tirando proveito de configurações padrão, quando possível.

Devido às configurações padrão, a API resultante é otimizada para fronteiras. Uma alternativa é [configurar uma API regional \(p. 197\)](#). Para configurar uma API regional, você deve definir explicitamente o tipo de endpoint da API como REGIONAL. Para configurar uma API otimizada para fronteiras explicitamente, você pode definir EDGE como o tipo do endpointConfiguration.

Ao configurar uma API, você deve escolher uma região. Quando implantada, a API é específica da região. Para uma API otimizada para fronteiras, o URL base tem o formato `http[s]://{{restapi-id}}.execute-api.amazonaws.com/stage`, em que `{restapi-id}` é o valor do id da API gerado pelo API Gateway. Você pode atribuir um nome de domínio personalizado (por exemplo, `apis.example.com`) como o nome de host da API e chamar a API com uma URL base no formato `https://apis.example.com/myApi`.

Tópicos

- [Configurar uma API usando o console do API Gateway \(p. 184\)](#)
- [Configurar uma API otimizada para fronteiras usando comandos da AWS CLI \(p. 184\)](#)
- [Configurar uma API otimizada para fronteiras usando o AWS SDK para Node.js \(p. 189\)](#)
- [Configurar uma API otimizada para fronteiras importando definições do OpenAPI \(p. 196\)](#)
- [Configurar uma API regional no API Gateway \(p. 197\)](#)
- [Criar uma API privada no Amazon API Gateway \(p. 200\)](#)

Configurar uma API usando o console do API Gateway

Para configurar uma API do API Gateway usando o console do API Gateway, consulte [TUTORIAL: Criar uma API com integração não proxy HTTP \(p. 59\)](#).

Aprenda a configurar uma API seguindo um exemplo. Para obter mais informações, consulte [TUTORIAL: Crie uma API REST importando um exemplo \(p. 45\)](#).

Como alternativa, você pode configurar uma API usando o recurso [Importar API \(p. 356\)](#) do API Gateway para fazer upload de uma definição de API externa, conforme expresso no [OpenAPI 2.0](#) com o [Extensões do API Gateway para OpenAPI \(p. 606\)](#). O exemplo fornecido em [TUTORIAL: Crie uma API REST importando um exemplo \(p. 45\)](#) usa o recurso Import API.

Configurar uma API otimizada para fronteiras usando comandos da AWS CLI

A configuração de uma API usando a AWS CLI exige trabalhar com os comandos `create-rest-api`, `create-resource` ou `get-resources`, `put-method`, `put-method-response`, `put-integration` e `put-integration-response`. Os procedimentos a seguir mostram como trabalhar com esses comandos da AWS CLI para criar a API PetStore simples do tipo de integração HTTP.

Para criar uma API PetStore simples usando a AWS CLI

1. Chame o comando `create-rest-api` para configurar a `RestApi` em uma região específica (`us-west-2`).

```
aws apigateway create-rest-api --name 'Simple PetStore (AWS CLI)' --region us-west-2
```

Veja a seguir a saída desse comando:

```
{  
    "name": "Simple PetStore (AWS CLI)",  
    "id": "vaz7da96z6",  
    "createdDate": 1494572809  
}
```

Anote o `id` retornado da `RestApi` recém-criada. Ele será necessário para a configuração de outras partes da API.

2. Chame o comando `get-resources` para recuperar o identificador de recurso raiz da `RestApi`.

```
aws apigateway get-resources --rest-api-id vaz7da96z6 --region us-west-2
```

Veja a seguir a saída desse comando:

```
{  
    "items": [  
        {  
            "path": "/",  
            "id": "begaltmsm8"  
        }  
    ]  
}
```

Anote o `Id` do recurso raiz. Ele é necessário para iniciar a definição da árvore de recursos de API e configurar os métodos e as integrações.

3. Chame o comando `create-resource` para anexar um recurso filho (`pets`) sob o recurso raiz (`begaltmsm8`):

```
aws apigateway create-resource --rest-api-id vaz7da96z6 \  
    --region us-west-2 \  
    --parent-id begaltmsm8 \  
    --path-part pets
```

Veja a seguir a saída desse comando:

```
{  
    "path": "/pets",  
    "pathPart": "pets",  
    "id": "6sxz2j",  
    "parentId": "begaltmsm8"  
}
```

Para anexar um recurso filho sob a raiz, especifique o `Id` do recurso raiz como o valor da propriedade `parentId`. Da mesma forma, para anexar um recurso filho sob o recurso `pets`, repita as etapas anteriores substituindo o valor `parent-id` pelo `pets` recurso `id` de `6sxz2j`:

```
aws apigateway create-resource --rest-api-id vaz7da96z6 \
--region us-west-2 \
--parent-id 6sxz2j \
--path-part '{petId}'
```

Para tornar uma parte do caminho um parâmetro de caminho, coloque-a entre chaves. Se houver êxito, o comando gerará a seguinte resposta:

```
{
  "path": "/pets/{petId}",
  "pathPart": "{petId}",
  "id": "rjkmth",
  "parentId": "6sxz2j"
}
```

Agora que você criou dois recursos: /pets (6sxz2j) e /pets/{petId} (rjkmth), você pode configurar os métodos neles.

4. Chame o comando `put-method` para adicionar o método HTTP GET ao recurso /pets. Isso cria um Method de API de GET /pets com acesso aberto, fazendo referência ao recurso /pets por seu valor de ID de 6sxz2j.

```
aws apigateway put-method --rest-api-id vaz7da96z6 \
--resource-id 6sxz2j \
--http-method GET \
--authorization-type "NONE" \
--region us-west-2
```

Veja a seguir a saída bem-sucedida desse comando:

```
{
  "apiKeyRequired": false,
  "httpMethod": "GET",
  "authorizationType": "NONE"
}
```

O método é para acesso aberto porque `authorization-type` é definido 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 “Controlar e gerenciar acesso a uma API REST” \(p. 361\)](#).

Para ativar o acesso de leitura ao recurso /pets/{petId} (rjkmth), adicione o método HTTP GET para criar um Method de API de GET /pets/{petId} da seguinte forma.

```
aws apigateway put-method --rest-api-id vaz7da96z6 \
--resource-id rjkmth --http-method GET \
--authorization-type "NONE" \
--region us-west-2 \
--request-parameters method.request.path.petId=true
```

Veja a seguir a saída bem-sucedida desse comando:

```
{
  "apiKeyRequired": false,
  "httpMethod": "GET",
  "authorizationType": "NONE",
```

```
    "requestParameters": {  
        "method.request.path.petId": true  
    }  
}
```

Observe que o parâmetro de caminho de solicitação de método de `petId` deve ser especificado como um parâmetro de solicitação obrigatório para seu valor definido dinamicamente ser mapeado para um parâmetro de solicitação de integração correspondente e transmitido para o back-end.

5. Chame o comando `put-method-response` para configurar a resposta 200 OK do método `GET /pets` especificando o recurso `/pets` por seu valor de ID de recurso `6sxz2j`.

```
aws apigateway put-method-response --rest-api-id vaz7da96z6 \  
--resource-id 6sxz2j --http-method GET \  
--status-code 200 --region us-west-2
```

Veja a seguir a saída desse comando:

```
{  
    "statusCode": "200"  
}
```

De forma semelhante, para definir a resposta 200 OK do método `GET /pets/{petId}`, faça o seguinte, especificando o recurso `/pets/{petId}` por seu valor de ID de recurso de `rjkmth`:

```
aws apigateway put-method-response --rest-api-id vaz7da96z6 \  
--resource-id rjkmth --http-method GET \  
--status-code 200 --region us-west-2
```

Depois de definir uma interface de cliente simples para a API, você poderá configurar a integração dos métodos de API com o back-end.

6. Chame o comando `put-integration` para configurar um `Integration` com um endpoint HTTP especificado para o método `GET /pets`. O recurso `/pets` é identificado por seu ID de recurso `6sxz2j`:

```
aws apigateway put-integration --rest-api-id vaz7da96z6 \  
--resource-id 6sxz2j --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:

```
{  
    "httpMethod": "GET",  
    "passthroughBehavior": "WHEN_NO_MATCH",  
    "cacheKeyParameters": [],  
    "type": "HTTP",  
    "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets",  
    "cacheNamespace": "6sxz2j"  
}
```

Observe que o `uri` de integração de `http://petstore-demo-endpoint.execute-api.com/petstore/pets` especifica o endpoint de integração do método `GET /pets`.

Da mesma forma, veja a seguir como criar uma solicitação de integração para o método `GET /pets/{petId}`:

```
aws apigateway put-integration \
--rest-api-id vaz7da96z6 \
--resource-id rjkmth \
--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
```

Aqui, o endpoint de integração, o uri de `http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}`, também usa um parâmetro de caminho (`id`). Seu valor é mapeado do parâmetro de caminho de solicitação de método correspondente de `{petId}`. O mapeamento é definido como parte do `request-parameters`. Se esse mapeamento não for definido aqui, o cliente receberá uma resposta de erro ao tentar chamar o método.

Veja a seguir a saída desse comando:

```
{
  "passthroughBehavior": "WHEN_NO_MATCH",
  "cacheKeyParameters": [],
  "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}",
  "httpMethod": "GET",
  "cacheNamespace": "rjkmth",
  "type": "HTTP",
  "requestParameters": {
    "integration.request.path.id": "method.request.path.petId"
  }
}
```

7. Chame o comando `put-integration-response` para criar um `IntegrationResponse` do método `GET /pets` integrado a um back-end HTTP.

```
aws apigateway put-integration-response --rest-api-id vaz7da96z6 \
--resource-id 6szx2j --http-method GET \
--status-code 200 --selection-pattern "" \
--region us-west-2
```

Veja a seguir a saída desse comando:

```
{
  "selectionPattern": "",
  "statusCode": "200"
}
```

Da mesma forma, chame o comando `put-integration-response` a seguir para criar uma `IntegrationResponse` do método `GET /pets/{petId}`:

```
aws apigateway put-integration-response --rest-api-id vaz7da96z6 \
--resource-id rjkmth --http-method GET \
--status-code 200 --selection-pattern "" \
--region us-west-2
```

Com as etapas anteriores, você concluiu a configuração de uma API simples que permite aos seus clientes consultar os animais de estimação disponíveis no site PetStore e visualizar um animal

individual de um identificador especificado. Para que ela possa ser chamada pelo seu cliente, você deve implantar a API.

8. Implante a API em um estágio stage, por exemplo, chamando `create-deployment`:

```
aws apigateway create-deployment --rest-api-id vaz7da96z6 \
--region us-west-2 \
--stage-name test \
--stage-description 'Test stage' \
--description 'First deployment'
```

Você pode testar essa API digitando a URL `https://vaz7da96z6.execute-api.us-west-2.amazonaws.com/test/pets` em um navegador e substituindo `vaz7da96z6` pelo identificador da sua API. A saída esperada deve ser a seguinte:

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

Para testar o método GET `/pets/{petId}`, digite `https://vaz7da96z6.execute-api.us-west-2.amazonaws.com/test/pets/3` no navegador. Você deve receber a seguinte resposta:

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

Configurar uma API otimizada para fronteiras usando o AWS SDK para Node.js

Como ilustração, usamos o AWS SDK para Node.js para descrever como usar um AWS SDK para criar uma API do API Gateway. Para obter mais informações de como usar um AWS SDK, incluindo como configurar o ambiente de desenvolvimento, consulte [AWS SDKs](#).

A configuração de uma API usando o AWS SDK para Node.js envolve chamar as funções `createRestApi`, `createResource` ou `getResources`, `putMethod`, `putMethodResponse`, `putIntegration` e `putIntegrationResponse`.

Os procedimentos a seguir mostram as etapas essenciais para usar esses comandos de SDK para configurar uma API PetStore simples oferecendo suporte aos métodos GET `/pets` e GET `/pets/{petId}`.

Para configurar uma API PetStore simples usando o AWS SDK para Node.js

1. Instancie o SDK:

```
var AWS = require('aws-sdk');

AWS.config.region = 'us-west-2';
var apig = new AWS.APIGateway({apiVersion: '2015/07/09'});
```

2. Chame a função `createRestApi` para configurar a entidade `RestApi`.

```
apig.createRestApi({
  name: "Simple PetStore (node.js SDK)",
  binaryMediaTypes: [
    '*'
  ],
  description: "Demo API created using the AWS SDK for node.js",
  version: "0.00.001"
}, function(err, data){
  if (!err) {
    console.log(data);
  } else {
    console.log('Create API failed:\n', err);
  }
});
```

A função retorna uma saída semelhante ao seguinte resultado:

```
{
  id: 'iupo308uaq7',
  name: 'PetStore (node.js SDK)',
  description: 'Demo API created using the AWS SDK for node.js',
  createdDate: 2017-09-05T19:32:35.000Z,
  version: '0.00.001',
  binaryMediaTypes: [ '*' ]}
```

O identificador da API resultante é `iupo308uaq7`. Você precisa fornecer isso para continuar a configuração da API.

3. Chame a função `getResources` para recuperar o identificador de recurso raiz da `RestApi`.

```
apig.getResources({
  restApiId: 'iupo308uaq7'
}, function(err, data){
  if (!err) {
    console.log(data);
  } else {
    console.log('Get the root resource failed:\n', err);
  }
});
```

Essa função retorna uma saída semelhante ao seguinte resultado:

```
{
  "items": [
    {
      "path": "/",
      "id": "s4fb0trnk0"
    }
  ]
}
```

```
}
```

O identificador de recurso raiz é s4fb0trnk0. Esse é o ponto de partida para construir a árvore de recursos de API, o que você verá em seguida.

4. Chame a função `createResource` para configurar o recurso `/pets` para a API, especificando o identificador de recurso raiz (`s4fb0trnk0`) na propriedade `parentId`.

```
apig.createResource({  
  restApiId: 'iuo308uaq7',  
  parentId: 's4fb0trnk0',  
  pathPart: 'pets'  
}, function(err, data){  
  if (!err) {  
    console.log(data);  
  } else {  
    console.log("The '/pets' resource setup failed:\n", err);  
  }  
})
```

O resultado bem-sucedido é o seguinte:

```
{  
  "path": "/pets",  
  "pathPart": "pets",  
  "id": "8sxa2j",  
  "parentId": "s4fb0trnk0"  
}
```

Para configurar o recurso `/pets/{petId}`, chame a seguinte função `createResource`, especificando o recurso `/pets` recém-criado (`8sxa2j`) na propriedade `parentId`.

```
apig.createResource({  
  restApiId: 'iuo308uaq7',  
  parentId: '8sxa2j',  
  pathPart: '{petId}'  
}, function(err, data){  
  if (!err) {  
    console.log(data);  
  } else {  
    console.log("The '/pets/{petId}' resource setup failed:\n", err);  
  }  
})
```

O resultado bem-sucedido retorna o valor de `id` recém-criado:

```
{  
  "path": "/pets/{petId}",  
  "pathPart": "{petId}",  
  "id": "au5df2",  
  "parentId": "8sxa2j"  
}
```

Durante esse procedimento, consulte o recurso `/pets`, especificando seu ID de recurso de `8sxa2j` e o recurso `/pets/{petId}` especificando seu ID de recurso de `au5df2`.

5. Chame a função `putMethod` para adicionar o método HTTP `GET` no recurso `/pets` (`8sxa2j`). Isso configura o `GET /pets` Method com acesso aberto.

```
apig.putMethod({
  restApiId: 'iuo308uaq7',
  resourceId: '8sxa2j',
  httpMethod: 'GET',
  authorizationType: 'NONE'
}, function(err, data){
  if (!err) {
    console.log(data);
  } else {
    console.log("The 'GET /pets' method setup failed:\n", err);
  }
})
```

Essa função retorna uma saída semelhante ao seguinte resultado:

```
{
  "apiKeyRequired": false,
  "httpMethod": "GET",
  "authorizationType": "NONE"
}
```

Para adicionar o método HTTP GET ao recurso `/pets/{petId}` (`au5df2`), que configura o método de API de `GET /pets/{petId}` com acesso aberto, chame a função `putMethod` da seguinte forma.

```
apig.putMethod({
  restApiId: 'iuo308uaq7',
  resourceId: 'au5df2',
  httpMethod: 'GET',
  authorizationType: 'NONE',
  requestParameters: {
    "method.request.path.petId" : true
  }
}, function(err, data){
  if (!err) {
    console.log(data);
  } else {
    console.log("The 'GET /pets/{petId}' method setup failed:\n", err);
  }
})
```

Essa função retorna uma saída semelhante ao seguinte resultado:

```
{
  "apiKeyRequired": false,
  "httpMethod": "GET",
  "authorizationType": "NONE",
  "requestParameters": {
    "method.request.path.petId": true
  }
}
```

Você precisa definir a propriedade `requestParameters` como mostrado no exemplo anterior para mapear e passar o valor `petId` fornecido pelo cliente para o back-end.

6. Chame a função `putMethodResponse` para configurar uma resposta para o método `GET /pets`.

```
apig.putMethodResponse({
  restApiId: 'iuo308uaq7',
  resourceId: '8sxa2j',
```

```
    httpMethod: 'GET',
    statusCode: '200'
}, function(err, data){
if (!err) {
  console.log(data);
} else {
  console.log("Set up the 200 OK response for the 'GET /pets' method failed:\n", err);
}

})
```

Essa função retorna uma saída semelhante ao seguinte resultado:

```
{  
  "statusCode": "200"  
}
```

Para definir a resposta 200 OK do método `GET /pets/{petId}`, chame a função `putMethodResponse` especificando o identificador de recurso `/pets/{petId}` (`au5df2`) na propriedade `resourceId`.

```
apig.putMethodResponse({
  restApiId: 'iuo308uaq7',
  resourceId: "au5df2",
  httpMethod: 'GET',
  statusCode: '200'
}, function(err, data){
if (!err) {
  console.log(data);
} else {
  console.log("Set up the 200 OK response for the 'GET /pets/{petId}' method failed:\n", err);
}

})
```

7. Chame a função `putIntegration` para configurar a `Integration` com um endpoint HTTP especificado para o método `GET /pets`, fornecendo o identificador de recurso `/pets` (`8sxa2j`) na propriedade `parentId`.

```
apig.putIntegration({
  restApiId: 'iuo308uaq7',
  resourceId: '8sxa2j',
  httpMethod: 'GET',
  type: 'HTTP',
  integrationHttpMethod: 'GET',
  uri: 'http://perstore-demo-endpoint.execute-api.com/pets'
}, function(err, data){
if (!err) {
  console.log(data);
} else {
  console.log("Set up the integration of the 'GET /' method of the API failed:\n", err);
}

})
```

Essa função retorna uma saída semelhante à seguinte:

```
{
```

```
        "httpMethod": "GET",
        "passthroughBehavior": "WHEN_NO_MATCH",
        "cacheKeyParameters": [],
        "type": "HTTP",
        "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
        "cacheNamespace": "8sxa2j"
    }
```

Para configurar a integração do método GET /pets/{petId} com o endpoint HTTP de http://petstore-demo-endpoint.execute-api.com/pets/{id} do back-end, chame a seguinte função `putIntegration`, fornecendo o identificador de recurso /pets/{petId} da API (au5df2) na propriedade `parentId`.

```
apig.putIntegration({
    restApiId: 'iuo308uaq7',
    resourceId: 'au5df2',
    httpMethod: 'GET',
    type: 'HTTP',
    integrationHttpMethod: 'GET',
    uri: 'http://petstore-demo-endpoint.execute-api.com/pets/{id}',
    requestParameters: {
        "integration.request.path.id": "method.request.path.petId"
    }
}, function(err, data){
    if (!err) {
        console.log(data);
    } else {
        console.log("The 'GET /pets/{petId}' method integration setup failed:\n", err);
    }
})
```

Essa função retorna uma saída bem-sucedida semelhante ao seguinte:

```
{
    "httpMethod": "GET",
    "passthroughBehavior": "WHEN_NO_MATCH",
    "cacheKeyParameters": [],
    "type": "HTTP",
    "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}",
    "cacheNamespace": "au5df2",
    "requestParameters": {
        "integration.request.path.id": "method.request.path.petId"
    }
}
```

8. Chame a função `putIntegrationResponse` para configurar a resposta de integração 200 OK para o método GET /pets especificando o identificador de recurso /pets da API (8sxa2j) na propriedade `resourceId`.

```
apig.putIntegrationResponse({
    restApiId: 'iuo308uaq7',
    resourceId: '8sxa2j',
    httpMethod: 'GET',
    statusCode: '200',
    selectionPattern: ''
}, function(err, data){
    if (!err) {
        console.log(data);
    } else {
        console.log("The 'GET /pets' method integration response setup failed:\n", err);
    }
})
```

```
})
```

Essa função retornará uma saída semelhante ao seguinte resultado:

```
{
  "selectionPattern": "",
  "statusCode": "200"
}
```

Para configurar a resposta de integração 200 OK do método GET /pets/{petId}, chame a função `putIntegrationResponse` especificando o identificador de recurso /pets/{petId} da API (au5df2) na propriedade `resourceId`.

```
apig.putIntegrationResponse({
  restApiId: 'iuo308uaq7',
  resourceId: 'au5df2',
  httpMethod: 'GET',
  statusCode: '200',
  selectionPattern: ''
}, function(err, data){
  if (!err) {
    console.log(data);
  } else {
    console.log("The 'GET /pets/{petId}' method integration response setup failed:\n",
    err);
  }
})
```

9. Como boa prática, experimente chamar a API antes de implantá-la. Para tentar invocar o método GET /pets, chame o `testInvokeMethod` especificando o identificador de recurso /pets (8sxa2j) na propriedade `resourceId`:

```
apig.testInvokeMethod({
  restApiId: 'iuo308uaq7',
  resourceId: '8sxa2j',
  httpMethod: "GET",
  pathWithQueryString: '/'
}, function(err, data){
  if (!err) {
    console.log(data)
  } else {
    console.log('Test-invoke-method on 'GET /pets' failed:\n', err);
  }
})
```

Para tentar invocar o método GET /pets/{petId}, chame o `testInvokeMethod` especificando o identificador de recurso /pets/{petId} (au5df2) na propriedade `resourceId`:

```
apig.testInvokeMethod({
  restApiId: 'iuo308uaq7',
  resourceId: 'au5df2',
  httpMethod: "GET",
  pathWithQueryString: '/'
}, function(err, data){
  if (!err) {
    console.log(data)
  } else {
    console.log('Test-invoke-method on 'GET /pets/{petId}' failed:\n', err);
  }
})
```

10. Por fim, você pode implantar a API para seus clientes chamarem.

```
apig.createDeployment({
  restApiId: 'iuo308uaq7',
  stageName: 'test',
  stageDescription: 'test deployment',
  description: 'API deployment'
}, function(err, data){
  if (err) {
    console.log('Deploying API failed:\n', err);
  } else {
    console.log("Deploying API succeeded\n", data);
  }
})
```

Configurar uma API otimizada para fronteiras importando definições do OpenAPI

Você pode configurar uma API no API Gateway especificando definições do OpenAPI de entidades de API do API Gateway apropriadas e importando as definições do OpenAPI para o API Gateway.

As seguintes definições do OpenAPI descrevem a API simples, expondo apenas o método GET / integrado a um endpoint HTTP do site PetStore no back-end e retornando uma resposta 200 OK.

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "title": "Simple PetStore (OpenAPI)"
  },
  "schemes": [
    "https"
  ],
  "paths": {
    "/pets": {
      "get": {
        "responses": {
          "200": {
            "description": "200 response"
          }
        },
        "x-amazon-apigateway-integration": {
          "responses": {
            "default": {
              "statusCode": "200"
            }
          },
          "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
          "passthroughBehavior": "when_no_match",
          "httpMethod": "GET",
          "type": "http"
        }
      }
    },
    "/pets/{petId)": {
      "get": {
        "parameters": [
          {
            "name": "petId",
            "in": "path",
          }
        ],
        "responses": {
          "200": {
            "description": "200 response"
          }
        }
      }
    }
  }
}
```

```
        "required": true,
        "type": "string"
    },
],
"responses": {
    "200": {
        "description": "200 response"
    }
},
"x-amazon-apigateway-integration": {
    "responses": {
        "default": {
            "statusCode": "200"
        }
    },
    "requestParameters": {
        "integration.request.path.id": "method.request.path.petId"
    },
    "uri": "http://petstore-demo-endpoint.execute-api.com/petstore/pets/{id}",
    "passthroughBehavior": "when_no_match",
    "httpMethod": "GET",
    "type": "http"
}
}
}
}
```

O procedimento a seguir descreve como importar essas definições do OpenAPI no API Gateway usando o console do API Gateway.

Para importar as definições do OpenAPI simples usando o console do API Gateway

1. Faça login no console do API Gateway.
2. Escolha Create API (Criar API).
3. Escolha Importar do OpenAPI.
4. Se você salvou as definições anteriores do OpenAPI em um arquivo, escolha Selecionar arquivo do OpenAPI. Você também pode copiar as definições do OpenAPI e colá-las no editor de texto de importação.
5. Selecione Import (Importar) para terminar de importar as definições do OpenAPI.

Para importar as definições do OpenAPI usando a AWS CLI, salve as definições do OpenAPI em um arquivo e depois execute o comando a seguir, supondo que você use a região us-west-2 e que o caminho de arquivo absoluto do OpenAPI seja file:///path/to/API_OpenAPI_template.json:

```
aws apigateway import-rest-api --body 'file:///path/to/API_OpenAPI_template.json' --region us-west-2
```

Configurar uma API regional no API Gateway

Quando as solicitações de API predominantemente se originam de uma instância do EC2 ou serviços dentro da mesma região em que a API está implantada, um endpoint de API regional, normalmente, reduzirá a latência de conexões e é recomendado para esses cenários.

Note

Nos casos em que os clientes da API são geograficamente dispersos, ainda poderá fazer sentido usar um endpoint de API regional, juntamente com 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?](#).

Para criar uma API regional, siga as etapas em [criar uma API otimizada para fronteiras \(p. 184\)](#), mas defina explicitamente o tipo REGIONAL como a única opção de endpointConfiguration da API.

A seguir, mostramos como criar uma API regional usando o console do API Gateway, a AWS CLI e o AWS SDK para Javascript para Node.js.

Tópicos

- [Criar uma API regional usando o console do API Gateway \(p. 198\)](#)
- [Criar uma API regional usando a AWS CLI \(p. 198\)](#)
- [Criar uma API regional usando o AWS SDK para JavaScript \(p. 199\)](#)
- [Testar uma API regional \(p. 199\)](#)

Criar uma API regional usando o console do API Gateway

Para criar uma API regional usando o console do API Gateway

1. Faça login no console do API Gateway e escolha + Create API (+ Criar API).
2. Em Create new API (Criar nova API), escolha a opção New API (Nova API).
3. Digite um nome (por exemplo, Simple PetStore (Console, Regional)) para API name (Nome da API).
4. Escolha Regional para Tipo de endpoint.
5. Escolha Create API (Criar API).

A partir de agora, você poderá configurar métodos de API e suas integrações associadas conforme descrito em [criar uma API otimizada para fronteiras \(p. 184\)](#).

Criar uma API regional usando a AWS CLI

Para criar uma API regional usando a AWS CLI, chame o comando `create-rest-api`:

```
aws apigateway create-rest-api \
    --name 'Simple PetStore (AWS CLI, Regional)' \
    --description 'Simple regional PetStore API' \
    --region us-west-2 \
    --endpoint-configuration '{ "types": [ "REGIONAL" ] }'
```

Uma resposta bem-sucedida retorna uma carga similar à seguinte:

```
{
  "createdDate": "2017-10-13T18:41:39Z",
  "description": "Simple regional PetStore API",
  "endpointConfiguration": {
    "types": "REGIONAL"
  },
  "id": "0qzs2sy7bh",
  "name": "Simple PetStore (AWS CLI, Regional)"
}
```

A partir de agora, você pode seguir as mesmas instruções fornecidas em [the section called “Configurar uma API otimizada para fronteiras usando comandos da AWS CLI” \(p. 184\)](#) para configurar métodos e integrações para essa API.

Criar uma API regional usando o AWS SDK para JavaScript

Para criar uma API regional usando o AWS SDK para JavaScript:

```
apig.createRestApi({
  name: "Simple PetStore (node.js SDK, regional)",
  endpointConfiguration: {
    types: ['REGIONAL']
  },
  description: "Demo regional API created using the AWS SDK for node.js",
  version: "0.00.001"
}, function(err, data){
  if (!err) {
    console.log('Create API succeeded:\n', data);
    restApiId = data.id;
  } else {
    console.log('Create API failed:\n', err);
  }
});
```

Uma resposta bem-sucedida retorna uma carga similar à seguinte:

```
{
  "createdDate": "2017-10-13T18:41:39Z",
  "description": "Demo regional API created using the AWS SDK for node.js",
  "endpointConfiguration": {
    "types": "REGIONAL"
  },
  "id": "0qzs2sy7bh",
  "name": "Simple PetStore (node.js SDK, regional)"
}
```

Depois que concluir as etapas anteriores, você poderá seguir as mesmas instruções em [the section called “Configurar uma API otimizada para fronteiras usando o AWS SDK para Node.js” \(p. 189\)](#) para configurar métodos e integrações para essa API.

Testar uma API regional

Uma vez implantada, o nome de host da URL padrão da API regional está no seguinte formato:

```
{restapi-id}.execute-api.{region}.amazonaws.com
```

A URL de base para chamar a API é a seguinte:

```
https://{restapi-id}.execute-api.{region}.amazonaws.com/{stage}
```

Supondo que você configure os métodos GET /pets e GET /pets/{petId} neste exemplo, você pode testar a API digitando as seguintes URLs em um navegador:

```
https://0qzs2sy7bh.execute-api.us-west-2.amazonaws.com/test/pets
```

e

```
https://0qzs2sy7bh.execute-api.us-west-2.amazonaws.com/test/pets/1
```

É possível também usar os comandos do cURL:

```
curl -X GET https://0qzs2sy7bh.execute-api.us-west-2.amazonaws.com/test/pets
```

e

```
curl -X GET https://0qzs2sy7bh.execute-api.us-west-2.amazonaws.com/test/pets/2
```

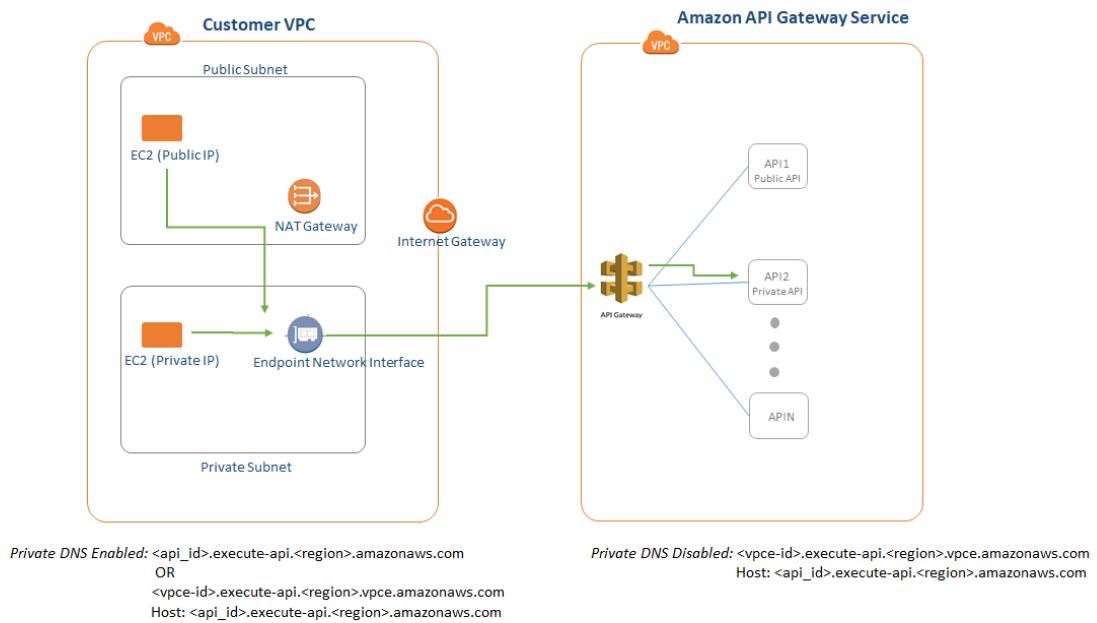
Criar uma API privada no Amazon API Gateway

Usando o Amazon API Gateway, você pode criar APIs REST privadas que só podem ser acessadas da 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. Com as [políticas de recursos \(p. 204\)](#), 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 AWS Direct Connect para estabelecer uma conexão a partir de uma rede local para a Amazon VPC 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.

Você pode [acessar \(p. 584\)](#) suas APIs privadas por meio dos VPC endpoints de interface do API Gateway conforme mostrado no diagrama a seguir. Se o DNS privado está habilitado, você pode usar Nomes de DNS público ou privado para acessar as APIs. Se o DNS privado está desabilitado, você só pode usar Nomes de DNS públicos.

Note

As APIs privadas do API Gateway são compatíveis somente com TLS 1.2. Versões anteriores do TLS não são compatíveis.



De maneira detalhada, as etapas para a criação de uma API privada são:

1. Primeiro, [crie um VPC endpoint de interface \(p. 201\)](#) para o serviço de componente do API Gateway a fim de executar a API, conhecido como `execute-api` na VPC.
2. Crie e teste sua API privada.

- a. Use um dos procedimentos a seguir para criar a API:
 - [Console do API Gateway \(p. 202\)](#)
 - [CLI do API Gateway \(p. 202\)](#)
 - [SDK do AWS para JavaScript \(p. 203\)](#)
- b. Para conceder acesso ao VPC endpoint, [crie uma política de recursos e a anexe à API \(p. 204\)](#).
- c. [Teste sua API \(p. 584\)](#).

Note

Os procedimentos a seguir pressupõem que você já tenha uma VPC totalmente configurada. Para obter mais informações e para começar a criar uma VPC, consulte [Conceitos básicos do Amazon VPC](#) no Guia do usuário da Amazon VPC.

Tópicos

- [Criar um VPC endpoint de interface para execute-api do API Gateway \(p. 201\)](#)
- [Criar uma API privada usando o console do API Gateway \(p. 202\)](#)
- [Criar uma API privada usando o console da AWS CLI \(p. 202\)](#)
- [Criar uma API privada usando o SDK da AWS para JavaScript \(p. 203\)](#)
- [Configurar uma política de recursos para uma API privada \(p. 204\)](#)
- [Implantar uma API privada usando o console do API Gateway \(p. 204\)](#)
- [Considerações sobre desenvolvimento da API privada \(p. 204\)](#)

Criar um VPC endpoint de interface para `execute-api` do API Gateway

O serviço de componente do API Gateway para a execução dessa API é chamado de `execute-api`. Para acessar sua API privada após a implantação, você precisará criar um VPC endpoint de interface para ela na sua VPC.

Assim que você criar seu VPC endpoint, você poderá usá-lo para acessar várias APIs privadas.

Para criar um VPC endpoint de interface `execute-api` do API Gateway

1. Faça login no console da Amazon VPC em <https://console.aws.amazon.com/vpc/>.
2. No painel de navegação, escolha Endpoints, Create Endpoint (Criar endpoint).
3. Para Service category (Categoria do serviço), selecione AWS services (Serviços da AWS).
4. Em Service Name (Nome do serviço), escolha o endpoint de serviço do API Gateway, incluindo a região à qual você deseja se conectar. Esse procedimento estará no formato `com.amazonaws.region.execute-api`, por exemplo, `com.amazonaws.us-east-1.execute-api`.

Para Type (Tipo), verifique se indica Interface.

5. Preencha as informações a seguir:
 - Em VPC, escolha a VPC na qual você deseja criar o endpoint.
 - Em Subnets (Sub-redes), selecione as sub-redes (zonas de disponibilidade) nas quais você deseja criar as interfaces de rede do endpoint.

Note

Pode não haver suporte para todas as zonas de disponibilidade de todos os serviços da AWS.

- Em Enable Private DNS Name (Habilitar nome DNS privado), deixe a caixa de seleção marcada. O DNS privado é habilitado por padrão.

Quando o DNS privado é habilitado, você poderá acessar sua API por DNS público ou privado. (Essa configuração não afeta quais usuários podem acessar sua API, somente quais endereços de DNS eles podem usar.) No entanto, não é possível acessar APIs públicas de uma VPC usando um VPC endpoint do API Gateway com o DNS privado habilitado. Observe que essas configurações de DNS não afetarão a capacidade de chamar essas APIs públicas da VPC se você estiver usando um nome de domínio personalizado otimizado para fronteiras para acessar a API pública. Usar um nome de domínio personalizado otimizado para fronteiras para acessar a API pública (ao usar DNS privado para acessar a API privada) é uma maneira de acessar as APIs públicas e privadas de uma VPC em que o endpoint foi criado com o DNS privado habilitado.

Note

Deixar o DNS privado habilitado é a opção recomendada. Se você optar por não habilitar o DNS privado, só poderá acessar sua API por meio do DNS público.

Para usar a opção de DNS privado, os atributos `enableDnsSupport` e `enableDnsHostnames` da sua VPC deverão ser definidos como `true`. Para obter mais informações, consulte o [Suporte a DNS na sua VPC](#) e a [Atualização do suporte a DNS da sua VPC](#) no Guia do usuário da Amazon VPC.

- 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.

6. Escolha Create endpoint.

Criar uma API privada usando o console do API Gateway

Como criar uma API privada usando o console do API Gateway

1. Faça login no console do API Gateway e escolha + Create API (+ Criar API).
2. Em Create new API (Criar nova API), escolha a opção New API (Nova API).
3. Digite um nome (por exemplo, `Simple PetStore (Console, Private)`) para API name (Nome da API).
4. Em Endpoint Type (Tipo de endpoint), selecione `Private`.
5. Escolha Create API (Criar API).

A partir de agora, você pode configurar métodos de API e suas integrações associadas conforme descrito nas etapas de 1 a 6 da sessão [???](#) (p. 60).

Note

Até que sua API tenha uma política de recursos que conceda acesso à [VPC ou ao VPC endpoint \(p. 201\)](#), todas as chamadas de API falharão. Antes de testar e implantar sua API, você precisará criar uma política de recursos e anexá-la à API conforme descrito em [???](#) (p. 204).

Criar uma API privada usando o console da AWS CLI

Para criar uma API privada usando a AWS CLI, chame o comando `create-rest-api`:

```
aws apigateway create-rest-api \
    --name 'Simple PetStore (AWS CLI, Private)' \
    --description 'Simple private PetStore API' \
```

```
--region us-west-2 \
--endpoint-configuration '{ "types": [ "PRIVATE" ] }'
```

Uma chamada bem-sucedida retorna uma saída semelhante ao seguinte:

```
{
  "createdDate": "2017-10-13T18:41:39Z",
  "description": "Simple private PetStore API",
  "endpointConfiguration": {
    "types": "PRIVATE"
  },
  "id": "0qzs2sy7bh",
  "name": "Simple PetStore (AWS CLI, Private)"
}
```

A partir de agora, você pode seguir as mesmas instruções fornecidas em [the section called “Configurar uma API otimizada para fronteiras usando comandos da AWS CLI” \(p. 184\)](#) para configurar métodos e integrações para essa API.

Assim que você estiver pronto para testar sua API, crie uma política de recursos e a anexe à API conforme descrito em [??? \(p. 204\)](#).

Criar uma API privada usando o SDK da AWS para JavaScript

Como criar uma API privada usando o SDK da AWS para JavaScript:

```
apig.createRestApi({
  name: "Simple PetStore (node.js SDK, private)",
  endpointConfiguration: {
    types: ['PRIVATE']
  },
  description: "Demo private API created using the AWS SDK for node.js",
  version: "0.00.001"
}, function(err, data){
  if (!err) {
    console.log('Create API succeeded:\n', data);
    restApiId = data.id;
  } else {
    console.log('Create API failed:\n', err);
  }
});
```

Uma chamada bem-sucedida retorna uma saída semelhante ao seguinte:

```
{
  "createdDate": "2017-10-13T18:41:39Z",
  "description": "Demo private API created using the AWS SDK for node.js",
  "endpointConfiguration": {
    "types": "PRIVATE"
  },
  "id": "0qzs2sy7bh",
  "name": "Simple PetStore (node.js SDK, private)"
}
```

Depois que concluir as etapas anteriores, você poderá seguir as mesmas instruções em [the section called “Configurar uma API otimizada para fronteiras usando o AWS SDK para Node.js” \(p. 189\)](#) para configurar métodos e integrações para essa API.

Assim que você estiver pronto para testar sua API, crie uma política de recursos e a anexe à API conforme descrito em [??? \(p. 204\)](#).

Configurar uma política de recursos para uma API privada

Antes de habilitar o acesso à sua API privada, você precisa criar uma política de recursos e anexá-la à API. Esse procedimento concederá acesso à API a partir de suas VPCs e VPC endpoints ou de VPCs e VPC endpoints em outras contas da AWS às quais você concedeu acesso explicitamente.

Para isso, siga as instruções em [the section called “Criar e anexar uma política de recursos do API Gateway para uma API” \(p. 375\)](#). Na etapa 4, escolha o exemplo da Source VPC Whitelist (Lista de permissões para VPC). Substitua `{}{vpceID}` (incluindo os colchetes) pelo ID do VPC endpoint e clique em Salvar para salvar sua política de recursos.

Considere também anexar uma política de endpoint ao VPC endpoint para especificar o acesso que está sendo concedido. Para obter mais informações, consulte [the section called “Usar políticas de VPC endpoint para APIs privadas” \(p. 394\)](#).

Implantar uma API privada usando o console do API Gateway

Para implantar sua API privada, siga estas instruções no console do API Gateway:

1. No painel de navegação à esquerda, selecione a API e depois escolha Deploy API (Implantar API) no menu suspenso Actions (Ações).
2. Na caixa de diálogo Deploy API (Implantar API), escolha um estágio (ou [New Stage] para a primeira implantação da API), insira um nome (por exemplo, "test", "prod", "dev" etc.) no campo de entrada Stage name (Nome do estágio), forneça opcionalmente uma descrição em Stage description (Descrição do estágio) e/ou Deployment description (Descrição da implantação) e, em seguida, escolha Deploy (Implantar).

Considerações sobre desenvolvimento da API privada

- Você pode converter uma API pública existente (regional ou otimizada para fronteiras) para uma API privada, bem como converter uma API privada para uma API regional. Não é possível converter uma API privada para uma API otimizada para fronteiras. Para obter mais informações, consulte [???](#) (p. 511).
- Para conceder acesso às VPCs e aos VPC endpoints à sua API privada, é necessário criar uma política de recursos e anexá-la à API recém-criada (ou convertida). Até que esse procedimento seja realizado, todas as chamadas para a API falharão. Para obter mais informações, consulte [???](#) (p. 204).
- Os [nomes de domínio personalizados](#) (p. 676) não são compatíveis com suas APIs privadas.
- Você pode usar um único VPC endpoint para acessar várias APIs privadas.
- Os VPC endpoints para APIs privadas estão sujeitos às mesmas limitações dos outros VPC endpoints de interface. Para obter mais informações, consulte [Limitações e propriedades do endpoint de interface](#) no Guia do usuário da Amazon VPC.
- É possível associar ou desassociar um VPC endpoint de uma API REST, o que fornece um registro DNS de alias do Route53 e simplifica a invocação de sua API privada. Para obter mais informações, consulte [Associar ou desassociar um VPC endpoint de uma API REST privada](#) (p. 515).

Configurar métodos de API 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 back-end 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 back-end, 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 \(p. 466\)](#) e [fornecer mensagens de erro adequadas \(p. 263\)](#) para [solicitações inválidas \(p. 343\)](#).

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 \(p. 45\)](#), 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 back-end. 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 back-end pode retornar. Na prática, você pode designar uma resposta de método como padrão para lidar com respostas desconhecidas ou não mapeadas do back-end. 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 é bem-sucedida no back-end.

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.

Tópicos

- [Configurar uma solicitação de método no API Gateway \(p. 205\)](#)
- [Configurar respostas de método no API Gateway \(p. 212\)](#)
- [Configurar um método usando o console do API Gateway \(p. 214\)](#)

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 \(p. 215\)](#)
- 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](#)).

Para ver exemplos de uso dessas ferramentas, consulte [Inicializar a configuração da API REST no API Gateway \(p. 184\)](#).

Tópicos

- [Configurar recursos da API \(p. 206\)](#)
- [Configurar um método HTTP \(p. 209\)](#)
- [Configurar parâmetros da solicitação de método \(p. 209\)](#)
- [Configurar o modelo de solicitação de método \(p. 210\)](#)
- [Configurar a autorização de solicitação de método \(p. 211\)](#)
- [Configurar a validação da solicitação de método \(p. 212\)](#)

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 Invoke URI (URI de invocação) e é exibido no editor de estágio 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 \(p. 676\)](#).

Para a [API de exemplo PetStore \(p. 45\)](#), 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 um 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.

Com a AWS CLI, você pode chamar o comando `get-resources` para descobrir quais recursos de uma API estão disponíveis:

```
aws apigateway get-resources --rest-api-id <apiId> \  
--region <region>
```

O resultado é uma lista dos recursos disponíveis da API no momento. Para o nosso exemplo de API da 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 o pai, chame o comando a seguir para adicionar um recurso filho.

```
aws apigateway create-resource --rest-api-id <apiId> \  
--region <region> \  
--parent-id <parentId> \  
--path-part <resourceName>
```

Por exemplo, para incluir alimento para animais de estimação no site PetStore, adicione um recurso `food` à raiz (/), definindo `path-part` como `food` e `parent-id` como `svzr2028x8`. O resultado parece o seguinte:

```
{  
    "path": "/food",  
    "pathPart": "food",  
    "id": "xdsvhp",  
    "parentId": "svzr2028x8",  
    "lastModified": "2018-03-20T13:45:00Z",  
    "resourceType": "METHOD",  
    "httpMethod": "GET",  
    "arn": "arn:aws:apigateway:us-east-1:svzr2028x8:method/get{/petId}"  
}
```

```
    "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 \(p. 222\)](#) 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/*. O nome de parâmetro de caminho voraz, proxy, pode ser substituído por outra string da mesma maneira que você trata um nome de parâmetro de caminho comum.

Usando a AWS CLI, chame o comando a seguir para configurar um recurso de proxy sob a raiz (/{{proxy+}}):

```
aws apigateway create-resource --rest-api-id <apiId> \  
  --region <region> \  
  --parent-id <rootResourceId> \  
  --path-part {proxy+}
```

O resultado é semelhante ao seguinte:

```
{  
  "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 back-end 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 back-end.

Uma API pode ter mais de um recurso de proxy. Por exemplo, os seguintes recursos de proxy são permitidos dentro de uma API.

```
/{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.

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 back-end 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 da AWS CLI a seguir mostra como criar uma solicitação de método do verbo [ANY](#) em um recurso especificado (`6sxz2j`), usando as permissões do IAM para controlar o acesso.

```
aws apigateway put-method --rest-api-id vaz7da96z6 \  
    --resource-id 6sxz2j \  
    --http-method ANY \  
    --authorization-type AWS_IAM \  
    --region us-west-2
```

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" \(p. 211\)](#).

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 back-end.

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. Isto é ilustrado da seguinte forma:

```
aws apigateway put-method --rest-api-id vaz7da96z6 \  
    --resource-id 6sxz2j \  
    --http-method ANY \  
    --method-parameters path.petId=12345
```

```
--resource-id rjkmth \
--http-method GET \
--authorization-type "NONE" \
--region us-west-2 \
--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 `breed`, você pode declará-los usando o seguinte comando da CLI, supondo-se que o recurso `/pets` id seja `6sxz2j`:

```
aws apigateway put-method --rest-api-id vaz7da96z6 \
  --resource-id 6sxz2j \
  --http-method GET \
  --authorization-type "NONE" \
  --region us-west-2 \
  --request-parameters
method.request.querystring.type=false,method.request.header.breed=false
```

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.breed":false}'
```

Com essa configuração, o cliente pode consultar animais de estimação por tipo:

```
GET /pets?type=dog
```

Além disso, o cliente pode consultar cães da raça poodle da seguinte forma:

```
GET /pets?type=dog
breed:poodle
```

Para obter informações sobre como mapear parâmetros da solicitação de método para parâmetros de solicitação de integração, consulte [the section called “Configurar integrações da API REST” \(p. 218\)](#).

Configurar o 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 \(p. 343\)](#), [gerar um SDK \(p. 548\)](#) e inicializar um modelo de mapeamento para configurar a integração no console do API Gateway. Para obter informações sobre como criar um [modelo](#), consulte [Modelos e modelos de mapeamento \(p. 273\)](#).

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. 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 `put-method` da AWS CLI.

Por exemplo, para definir um modelo na carga útil JSON da solicitação de método `POST /pets` da API de exemplo PetStore, você pode chamar o comando da AWS CLI a seguir:

```
aws apigateway put-method \
  --rest-api-id vaz7da96z6 \
```

```
--resource-id 6sxz2j \
--http-method POST \
--authorization-type "NONE" \
--region us-west-2 \
--request-models '{"application/json":"petModel"}'
```

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 a fim de 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 chamador na solicitação, com base na chave secreta e no identificador de chave de acesso do usuário do IAM. Se o usuário verificado tiver permissão para chamar o método, a solicitação será aceita. Caso contrário, a solicitação será rejeitada, e o chamador receberá uma resposta de erro não autorizada. A chamada para o método não será bem-sucedida, a menos que o chamador tenha recebido permissão para chamar o método de API ou se o chamador tiver permissão para assumir uma função que tenha recebido a permissão. O cliente tem permissões para chamar esse e quaisquer outros métodos de API criados por qualquer pessoa da mesma conta da AWS se o chamador tiver a seguinte política do IAM anexada ao seu usuário do IAM:

```
{
    "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” \(p. 378\)](#).

No momento, essa política pode ser concedida somente a usuários do IAM da conta do proprietário da API. Os usuários de outra conta da AWS poderão chamar os métodos de API se eles tiverem permissão para assumir uma função da conta do proprietário da API e a função assumida tiver as permissões adequadas para 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 SDK da AWS ou um cliente da API REST, como [Postman](#), que implementa a [assinatura do Signature versão 4](#).

Para usar um autorizador do Lambda a fim de 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 de propriedade `id` de um autorizador do Lambda que já exista. 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 autorizadores do Lambda” \(p. 396\)](#).

Para usar um grupo de usuários do Amazon Cognito para autorizar o acesso ao método da API, defina a propriedade de entrada `authorization-type` como COGNITO_USER_POOLS e defina a propriedade de entrada `authorizer-id` como o valor de propriedade `id` do autorizador COGNITO_USER_POOLS que já foi criado. Para obter informações sobre a criação de um autorizador do grupo de usuários do Amazon Cognito, consulte [the section called “Use o Grupo de usuários do Cognito como um autorizador para uma API REST” \(p. 414\)](#).

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](#):

```
aws apigateway create-request-validator \
  --rest-api-id 7zw9uyk9kl \
  --name bodyOnlyValidator \
  --validate-request-body \
  --no-validate-request-parameters
```

Esse comando da CLI cria um validador de solicitação somente de corpo. O exemplo de saída é o seguinte:

```
{
  "validateRequestParameters": true,
  "validateRequestBody": true,
  "id": "jgpwy6",
  "name": "bodyOnlyValidator"
}
```

Com esse validador de solicitação, você pode ativar a validação de solicitação como parte da configuração de solicitação de método:

```
aws apigateway put-method \
  --rest-api-id 7zw9uyk9kl
  --region us-west-2
  --resource-id xdsvhp
  --http-method PUT
  --authorization-type "NONE"
  --request-parameters '{"method.request.querystring.type": false,
"method.request.querystring.page":false}'
  --request-models '{"application/json":"petModel"}'
  --request-validator-id jgpwy6
```

Para ser incluído na validação de solicitação, um parâmetro de solicitação deve ser declarado, conforme necessário. Se o parâmetro de string de consulta para a página for usado na validação de solicitação, o mapa `request-parameters` do exemplo anterior deverá ser especificado como `'{"method.request.querystring.type": false,
"method.request.querystring.page":true}'`.

Configurar respostas 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 back-end para a resposta de método automaticamente. Não há necessidade de configurar a resposta de método de API. No entanto, com a integração de proxy do Lambda, a função do Lambda deve retornar um resultado [desse formato de saída \(p. 238\)](#) 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 redefiní-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 seguinte comando da AWS CLI cria uma resposta de método 200.

```
aws apigateway put-method-response \
    --region us-west-2 \
    --rest-api-id vaz7da96z6 \
    --resource-id 6sxz2j \
    --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 seguinte comando da CLI mostra um exemplo de definição do cabeçalho `my-header`, a variável de caminho `petId` e o parâmetro de consulta `query` como os destinos de mapeamento:

```
aws apigateway put-method-response \
    --region us-west-2 \
    --rest-api-id vaz7da96z6 \
    --resource-id 6sxz2j \
    --http-method GET \
    --status-code 200 \
    --response-parameters method.request.header.my-
header=false,method.request.path.petId=true,method.request.querystring.query=false
```

Configurar modelos de resposta de método

Um modelo de resposta de método define um formato do corpo de resposta de método. 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 exemplo a seguir mostra como criar um modelo PetStorePet para descrever o corpo da resposta para a solicitação do método GET /pets/{petId}.

```
aws apigateway create-model \
--region us-west-2 \
--rest-api-id vaz7da96z6 \
--content-type application/json \
--name PetStorePet \
--schema '{ \
    "$schema": "http://json-schema.org/draft-04/schema#", \
    "title": "PetStorePet", \
    "type": "object", \
    "properties": { \
        "id": { "type": "number" }, \
        "type": { "type": "string" }, \
        "price": { "type": "number" } \
    } \
}'
```

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 seguinte comando da AWS CLI de [put-method-response](#) mostra como isso é feito:

```
aws apigateway put-method-response \
--region us-west-2 \
--rest-api-id vaz7da96z6 \
--resource-id 6sxzz2j \
--http-method GET \
--status-code 200 \
--request-parameters method.request.header.my-
header=false,method.request.path.petId=true,method.request.querystring.query=false
--request-models '{"application/json":"PetStorePet"}'
```

A configuração de um modelo de resposta de método é necessária quando você gera um SDK fortemente tipado para a API. Ele garante que a saída seja convertida em uma classe adequada em Java ou Objective-C. Em outros casos, definir um modelo é opcional.

Configurar um método usando o console do API Gateway

Antes de configurar um método de API, verifique o seguinte:

- Você deve ter o método disponível no API Gateway. Siga as instruções em [TUTORIAL: Criar uma API com integração não proxy HTTP \(p. 59\)](#).
- Se você deseja que o método se comunique com uma função do Lambda, é necessário já ter criado a função de invocação do Lambda e a função de 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 a integração não proxy do Lambda \(p. 36\)](#) do [Criar uma API do API Gateway com integração à Lambda \(p. 28\)](#).
- 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 detalhes, consulte [Autoridades de certificado com suporte pelo API Gateway para integrações HTTP e de proxy HTTP \(p. 432\)](#).

Tópicos

- [Configurar uma solicitação de método do API Gateway no console do API Gateway \(p. 215\)](#)
- [Configurar uma resposta de método do API Gateway no console do API Gateway \(p. 217\)](#)

Configurar uma solicitação de método do API Gateway no console do API Gateway

Para usar o console do API Gateway para especificar a solicitação/resposta de método de uma API e configurar a maneira como o método autorizará solicitações, siga estas instruções.

Note

Estas instruções supõem que você já concluiu as etapas em [Configurar uma solicitação de integração de API usando o console do API Gateway \(p. 223\)](#). Elas são mais usadas para complementar as discussões apresentadas em [Criar uma API do API Gateway com integração à Lambda \(p. 28\)](#).

1. Com o método selecionado no painel Resources (Recursos), escolha Method Request (Solicitação de método) no painel Method Execution (Execução de método).
2. Em Settings (Configurações), escolha o ícone de lápis para abrir o menu suspenso Authorization (Autorização) e escolha um dos autorizadores disponíveis.
 - a. Para ativar o acesso aberto ao método para qualquer usuário, escolha NONE. Essa etapa poderá ser ignorada se a configuração padrão não tiver sido alterada.
 - b. Para usar permissões do IAM a fim de controlar o acesso do cliente ao método, escolha 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:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "execute-api:Invoke"  
            ],  
            "Resource": [  
                "resource-statement"  
            ]  
        }  
    ]  
}
```

Nessa política de acesso, *resource-statement* é o valor do campo ARN, na seção Authorization Settings (Configurações da autorização). Para obter mais informações sobre como definir as permissões do IAM, consulte [Controlar o acesso a uma API com permissões do IAM \(p. 378\)](#).

Para criar a função do IAM, você pode adaptar as instruções em "Para criar a função de invocação do Lambda e sua política" e "Para criar a função de execução do Lambda e sua política", na seção [Criar funções do Lambda \(p. 36\)](#) do [Criar uma API do API Gateway com integração à Lambda \(p. 28\)](#).

Para salvar sua opção, escolha Update (Atualizar). Caso contrário, escolha Cancel (Cancelar).

- c. Para usar um autorizador do Lambda, escolha um em Token authorizer (Autorizador de token). Você deve ter criado um autorizador do Lambda para ter essa opção exibida no menu suspenso. Para obter informações sobre como criar um autorizador do Lambda, consulte [Usar autorizadores do Lambda do API Gateway \(p. 396\)](#).
 - 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). Você deve ter criado 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 [Controle o acesso à API REST usando o Grupos de usuários do Amazon Cognito como um autorizador \(p. 414\)](#).
3. Para ativar ou desativar a validação de solicitação, escolha o ícone de lápis no menu suspenso Request Validator (Validador de solicitação) e escolha uma das opções listadas. Para obter mais informações sobre cada opção, consulte [Ativar a validação de solicitação no API Gateway \(p. 343\)](#).
 4. Para exigir uma chave de API, escolha o ícone de lápis para abrir o menu suspenso API Key Required (Chave da API necessária) e escolha true ou false de acordo com seus requisitos de API. Quando ativadas, as chaves de APIs são usadas em [planos de uso \(p. 450\)](#) para controlar o tráfego do cliente.
 5. Para adicionar um parâmetro de string de consulta ao método, faça o seguinte:
 - a. Escolha a seta ao lado de URL Query String Parameters (Parâmetros da string de consulta da URL) e escolha Add query string (Adicionar string de consulta).
 - b. Em Name (Nome), digite o nome do parâmetro de string de consulta.
 - c. Escolha o ícone de marca de seleção para salvar o novo nome de parâmetro de string de consulta.
 - d. Se o parâmetro de string de consulta recém-criado precisar ser usado para a validação de solicitação, escolha a opção Required (Obrigatório). Para obter mais informações sobre a validação de solicitação, consulte [Ativar a validação de solicitação no API Gateway \(p. 343\)](#).
 - e. Se o parâmetro de string de consulta recém-criado precisar ser usado como parte de uma chave de armazenamento em cache, marque a opção Caching (Armazenamento em cache). Isso é aplicável apenas quando o armazenamento em cache está ativado. Para obter mais informações sobre armazenamento em cache, consulte [Usar parâmetros de método/integração como chaves de cache \(p. 526\)](#).

Tip

Para remover o parâmetro de string de consulta, selecione o ícone x associado a ele e selecione Remove this parameter and any dependent parameters (Remover este parâmetro e todos os parâmetros dependentes) para confirmar a remoção.
Para alterar o nome do parâmetro de string de consulta, remova-o e crie um novo.

6. Para adicionar um parâmetro de cabeçalho ao método, faça o seguinte:
 - a. Escolha a seta ao lado de HTTP Request Headers (Cabeçalhos de solicitação HTTP) e escolha Add header (Adicionar cabeçalho).
 - b. Em Name (Nome), digite o nome do parâmetro de cabeçalho e, em seguida, escolha o ícone de marca de seleção para salvar as configurações.
 - c. Se o parâmetro de cabeçalho recém-criado precisar ser usado para a validação de solicitação, escolha a opção Required (Obrigatório). Para obter mais informações sobre a validação de solicitação, consulte [Ativar a validação de solicitação no API Gateway \(p. 343\)](#).
 - d. Se o parâmetro de cabeçalho recém-criado precisar ser usado como parte de uma chave de armazenamento em cache, escolha a opção Caching (Armazenamento em cache). Isso é aplicável apenas quando o armazenamento em cache está ativado. Para obter mais informações

sobre armazenamento em cache, consulte [Usar parâmetros de método/integração como chaves de cache \(p. 526\)](#).

Tip

Para remover o parâmetro de cabeçalho, escolha o ícone x associado a ele e, em seguida, escolha Remove this parameter and any dependent parameters (Remover este parâmetro e todos os parâmetros dependentes) para confirmar a remoção.

Para alterar o nome do parâmetro de cabeçalho, remova-o e crie um novo.

7. Para declarar o formato de carga útil de uma solicitação de método com um dos verbos HTTP `POST`, `PUT` ou `PATCH`, expanda Request Body (Solicitar corpo) e faça o seguinte:
 - a. Escolha Add model (Adicionar modelo).
 - b. Digite um tipo de MIME (por exemplo, `application/json`) para Content type (Tipo de conteúdo).
 - c. Abra o menu suspenso Model name (Nome do modelo) para escolher um modelo disponível para a carga útil e escolha o ícone de marca de seleção para salvar as configurações.

Os modelos atualmente disponíveis para a API incluem os modelos `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 [Criar um modelo \(p. 279\)](#).

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.

8. Para atribuir um nome de operação em um SDK do Java dessa API, gerada pelo API Gateway, expanda SDK Settings (Configurações do SDK) e digite um nome em Operation name (Nome da operação). Por exemplo, para a solicitação de método de `GET /pets/{petId}`, o nome da operação de Java SDK correspondente é, por padrão, `GetPetById`. 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`.

Configurar uma resposta de método do API Gateway no 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. Escolha Method Response (Resposta de método) em Method Execution (Execução de método) para um determinado método de recurso da API.
2. Para adicionar uma nova resposta, escolha Add Response (Adicionar resposta).
 - a. Digite um código de status HTTP; por exemplo, 200, 400 ou 500) para HTTP Status (Status HTTP) e, em seguida, escolha o ícone de marca de seleção para salvar a opção.

Quando uma resposta retornada pelo back-end 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`.

- b. Expanda a resposta do código de status determinado.
- c. Escolha Add Header (Adicionar cabeçalho).
- d. Digite um nome para Name (Nome) em Response Headers for `{status}` (Cabeçalhos de resposta para `{status}`) e, em seguida, escolha o ícone de marca de seleção para salvar a opção.

Se você precisar converter qualquer cabeçalho retornado pelo back-end em um definido em uma resposta de método, você deverá primeiro adicionar o cabeçalho de resposta de método, como descrito nesta etapa.

- e. Escolha Add Response Model (Adicionar modelo de resposta) em Response Body for `{status}` (Corpo da resposta para `{status}`).
 - f. Digite o tipo de mídia da carga útil de resposta para Content type (Tipo de conteúdo) e escolha um modelo no menu suspenso Models (Modelos).
 - g. Escolha o ícone de marca de seleção para salvar as configurações.
3. Para modificar uma resposta existente, expanda a resposta e siga a Etapa 2 acima.
 4. Para remover uma resposta, escolha o ícone x para a resposta e confirme que deseja excluir a resposta.

Para cada resposta retornada do back-end, 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 back-end 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.

Configurar integrações da API REST no API Gateway

Após a configuração de um método de API, você deve integrá-la a um endpoint no back-end. Um endpoint de back-end 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 back-end. 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 back-end.

A configuração de uma solicitação de integração envolve o seguinte: configurar como transmitir solicitações de método enviadas pelo cliente para o back-end; 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 o serviço da AWS a ser chamado.

A configuração de uma resposta de integração, aplicável a integrações não proxy, envolve o seguinte: configurar como transmitir o resultado retornado pelo back-end 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 do corpo especificado.

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. Em seguida, defina as permissões do IAM para acessar o back-end e especifique os

mapeamentos para transformar os dados da solicitação recebida antes de transmiti-los para o back-end. 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 back-end para a resposta de método.

Tópicos

- [Configurar uma solicitação de integração no API Gateway \(p. 219\)](#)
- [Configurar uma resposta de integração no API Gateway \(p. 226\)](#)
- [Configurar integrações do Lambda no API Gateway \(p. 227\)](#)
- [Configurar integrações HTTP no API Gateway \(p. 247\)](#)
- [Configurar integrações privadas do API Gateway \(p. 252\)](#)
- [Configurar integrações simuladas no API Gateway \(p. 260\)](#)

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 back-end.
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 back-end 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 back-end 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 fazê-lo usando o console do API Gateway, os comandos da AWS CLI, um SDK da AWS ou a API REST do API Gateway.

Tópicos

- [Tarefas básicas de uma solicitação de integração da API \(p. 219\)](#)
- [Escolher um tipo de integração da API do API Gateway \(p. 221\)](#)
- [Configurar a integração de proxy com um recurso de proxy \(p. 222\)](#)
- [Configurar uma solicitação de integração de API usando o console do API Gateway \(p. 223\)](#)

Tarefas básicas de uma solicitação de integração da API

Solicitação de integração é uma solicitação HTTP que o API Gateway envia para o back-end, transmitindo os dados da solicitação enviada pelo cliente e transformando os dados, se necessário. O método HTTP (ou o verbo) e o URI da solicitação de integração são determinados pelo back-end (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 obtido 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 [Configurar mapeamentos de dados de solicitação e resposta usando o console do API Gateway \(p. 270\)](#).

Ao configurar a solicitação de integração de proxy HTTP ou o HTTP, você atribui a URL de endpoint HTTP de back-end 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 de proxy do Lambda, você atribui o nome de recurso da Amazon (ARN) para invocar a função do Lambda como o valor do URI da solicitação de integração. 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 `Invoke` 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 uma existente para que o API Gateway chame a ação e, em seguida, especifique 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, em seguida, 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. Depois de discutir a configuração básica para uma resposta de integração, abordamos tópicos avançados em [Configurar mapeamentos de dados de solicitação e resposta usando o console do API Gateway \(p. 270\)](#), onde também abordamos a transmissão da carga útil e o processamento das codificações de conteúdo.

Escolher um tipo de integração da API do API Gateway

Você escolhe um tipo de integração da API de acordo com os tipos de endpoint de integração com os quais você trabalha e como você deseja transmitir dados de e para o endpoint de integração. Para uma função do Lambda, você pode ter a integração de 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 tem a integração da AWS somente do tipo não proxy. O API Gateway também oferece suporte à integração simulada, onde 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 de invocação de função](#) 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 de 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 back-end 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 back-end 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 passa a solicitação recebida do cliente como a entrada para a função do Lambda de back-end. A função integrada do Lambda obtém a [entrada desse formato \(p. 234\)](#) e analisa a entrada de todas as origens disponíveis, incluindo cabeçalhos de solicitação, variáveis de caminho do URL, parâmetros de string de consulta e o corpo aplicável. A função retorna o resultado seguindo esse [formato de saída \(p. 238\)](#). 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 que não sejam a de invocação de função.
- **HTTP:** Este tipo de integração permite que uma API exponha endpoints HTTP no back-end. Com a integração **HTTP**, também conhecida como integração personalizada HTTP, você deve configurar a solicitação de integração e a resposta de integração. Você deve 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.
- **HTTP_PROXY:** A integração de proxy HTTP permite que um cliente acesse os endpoints HTTP do back-end 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 back-end. 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 back-end 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 \(p. 263\)](#) 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, realize as tarefas a seguir:

- 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 back-end 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” \(p. 730\)](#).

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 back-end 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 back-end. Com esse tipo de integração, o API Gateway simplesmente passa toda solicitação e resposta entre o front-end e o back-end, sujeito a certas [restrições e limitações](#) (p. 730).

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 back-end HTTP com uma única configuração de integração. Por exemplo, suponha que o back-end 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 back-end 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 back-end.

[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 back-end. 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+}` a fim de configurar uma única integração para que uma função do Lambda de back-end reaja individualmente a alterações em qualquer um dos recursos da API em `/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 back-end 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.

O procedimento a seguir descreve como usar o console do API Gateway para especificar configurações de método.

1. No painel Resources (Recursos), escolha o método.
2. No painel Method Execution (Execução de método), escolha Solicitação de integração (Integration Request). Para Integration type (Tipo de integração), escolha uma das seguintes opções:
 - Escolha Lambda Function (Função do Lambda) se a sua API for integrada a uma função do Lambda. No nível da API, este é um tipo de integração AWS.
 - Escolha HTTP, se a sua API for integrada a um endpoint HTTP. No nível da API, este é o tipo de integração HTTP.
 - Escolha AWS Service (Serviço da AWS) se sua API for integrada diretamente a um serviço da AWS. No nível da API, este é o tipo de integração AWS. A opção Lambda Function (Função do Lambda) acima é um caso especial de integração da AWS para invocar uma função do Lambda e está disponível somente no console do API Gateway. Para configurar uma API do API Gateway para criar uma nova função do Lambda no AWS Lambda, a fim de definir uma permissão de recurso na função do Lambda ou executar qualquer outra ação de serviço do Lambda, você deve escolher a opção AWS Service (Serviço da AWS) aqui.
 - Escolha Mock (Simular) 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. Se você escolher essa opção, pule o restante das instruções neste tópico e consulte [Configurar integrações simuladas no API Gateway \(p. 260\)](#).
3. Se você escolheu Lambda Function (Função do Lambda), faça o seguinte:
 - a. Para Lambda Region (Região do Lambda), escolha o identificador de região que corresponde ao nome da região na qual você criou a função do Lambda. Por exemplo, se você tivesse criado a função do Lambda na região Leste dos EUA (Norte da Virgínia), escolheria **us-east-1**. Para obter uma lista de nomes e identificadores de região, consulte [AWS Lambda](#) na Referência geral do Amazon Web Services.
 - b. Para Lambda Function (Função do Lambda), digite o nome da função do Lambda e escolha o ARN correspondente da função.
 - c. Escolha Salvar.
4. Se você escolheu HTTP, faça o seguinte:
 - a. Em HTTP method (Método HTTP), escolha o tipo de método HTTP mais parecido com o método no back-end HTTP.
 - b. Em Endpoint URL (URL de endpoint), digite o URL do back-end HTTP que você deseja que esse método use.
 - c. Escolha Salvar.
5. Se você escolheu Mock (Simular), faça o seguinte:
 - Escolha Salvar.
6. Se você escolheu AWS Service (Serviço da AWS), faça o seguinte:
 - a. Para AWS Region (Região do AWS), escolha a região da AWS que esse método deve usar para chamar a ação.
 - b. Para AWS Service (Serviço da AWS), escolha o serviço da AWS que esse método deve chamar.
 - c. Para AWS Subdomain (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.

- d. 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 para AWS Service (Serviço da AWS).
- e. Para Action (Ação), digite a ação que você deseja usar. Para obter uma lista de ações disponíveis, consulte a documentação de referência da API para o serviço da AWS que você escolheu para AWS Service (Serviço da AWS).
- f. Em Execution Role (Função de execução), digite o ARN da função do IAM que o método usará para chamar a ação.

Para criar a função do IAM, você pode adaptar as instruções em "Para criar a função de invocação do Lambda e suas políticas" e "Para criar a função de execução do Lambda e sua política", na seção [Criar funções Lambda \(p. 36\)](#). Especifique uma política de acesso do seguinte formato, com o número desejado de instruções de recursos e ações:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "action-statement"  
            ],  
            "Resource": [  
                "resource-statement"  
            ]  
        },  
        ...  
    ]  
}
```

Para a sintaxe da instrução de ação e recurso, consulte a documentação do serviço da AWS que você escolheu para AWS Service (Serviço da AWS).

Para a relação de confiança da função do IAM, especifique o seguinte, o que permite que o API Gateway realize ações em nome da sua conta da AWS:

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

- g. Se a ação que você digitou para Action (Ação) fornecer um caminho de recurso personalizado que você deseja que esse método utilize, para Path Override (Substituição do caminho), digite esse caminho de recurso personalizado. Para o caminho de recurso personalizado, consulte a documentação de referência da API para o serviço da AWS que você escolheu para AWS Service (Serviço da AWS).
- h. Escolha Salvar.

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 back-end 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 back-end 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 back-end 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 back-end. Para um endpoint HTTP, a resposta do back-end é uma resposta HTTP. O código de status da resposta de integração pode obter o código de status retornado pelo back-end e o corpo da resposta de integração é a carga útil retornada pelo back-end. Para obter um endpoint do Lambda, a resposta do back-end é a saída retornada da 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 [Lidar com erros do Lambda no API Gateway \(p. 243\)](#). 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 [Criar modelos e modelos de mapeamento para mapeamentos de solicitação e resposta \(p. 273\)](#).

Configurar integrações do Lambda no API Gateway

Você pode integrar um método de API a uma função Lambda usando a integração de proxy do Lambda ou a integração não proxy (personalizada) do Lambda.

Na integração de proxy do Lambda, a configuração é simples. Se sua API não exigir codificação de conteúdo nem armazenamento em cache, você só precisará definir 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 a credencial como uma função do IAM com permissões para o API Gateway 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

- [Configurar integrações de proxy Lambda no API Gateway \(p. 227\)](#)
- [Configurar integrações personalizadas do Lambda no API Gateway \(p. 239\)](#)
- [Configurar a invocação assíncrona da função Lambda do back-end \(p. 243\)](#)
- [Lidar com erros do Lambda no API Gateway \(p. 243\)](#)

Configurar integrações de proxy Lambda no API Gateway

Tópicos

- [Compreender a integração de proxy do Lambda no API Gateway \(p. 227\)](#)
- [Suporte para cabeçalhos de vários valores e parâmetros de string de consulta \(p. 229\)](#)
- [Configurar um recurso de proxy com uma integração de proxy do Lambda \(p. 229\)](#)
- [Configurar a integração de proxy do Lambda usando a AWS CLI \(p. 231\)](#)
- [Formato de entrada de uma função Lambda para integração de proxy \(p. 234\)](#)
- [Formato de saída de uma função Lambda para integração de proxy \(p. 238\)](#)

Compreender a integração de proxy do Lambda no API Gateway

A integração de proxy do Amazon API Gateway Lambda 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 back-end. A função acessa muitos recursos ou recursos de outros serviços da AWS, incluindo chamadas para outras funções Lambda.

Na integração de proxy do Lambda, quando um cliente envia uma solicitação de API, o API Gateway repassa para a função do Lambda integrada a solicitação bruta no estado em que se encontra, exceto pelo fato de que a ordem dos parâmetros da solicitação não é preservada. Esses [dados de solicitação \(p. 234\)](#) 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 Lambda do back-end 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 \(p. 238\)](#).

Como o API Gateway não interfere muito entre o cliente e a função do Lambda do back-end 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 aplicativos promulgados pela função Lambda de back-end.

É possível configurar uma integração de proxy Lambda para qualquer método de API. Mas uma integração de proxy 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 Lambda de back-end 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 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 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 mais informações sobre como configurar integrações de proxy do API Gateway, consulte [Configurar a integração de proxy com um recurso de proxy \(p. 222\)](#).

A integração de proxy exige que o cliente tenha um conhecimento mais detalhado dos requisitos de back-end. Portanto, para garantir um melhor desempenho dos aplicativos e experiência do usuário, o desenvolvedor de back-end deve comunicar claramente ao desenvolvedor cliente os requisitos do back-end 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](#)

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 Lambda para integração de proxy \(p. 234\)](#) e [Formato de saída de uma função Lambda para integração de proxy \(p. 238\)](#).

[Configurar um recurso de proxy com uma integração de proxy do Lambda](#)

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 back-end da função 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 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 Lambda, em tempo de execução, o API Gateway mapeia uma solicitação de entrada para o parâmetro event de entrada da função 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 Lambda para integração de proxy \(p. 234\)](#). 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 Lambda para integração de proxy \(p. 238\)](#).

Na integração de proxy Lambda de um recurso de proxy por meio do método ANY, a função Lambda de back-end 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 Lambda de back-end pode analisar o caminho da URL e inserir as tuplas de localização em uma tabela do DynamoDB.

Configurar a integração de proxy do Lambda usando a AWS CLI

Nesta seção, mostramos como usar a AWS CLI para configurar uma API com a integração de proxy do Lambda.

Note

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: Criar uma API Hello World com integração de proxy do Lambda \(p. 29\)](#).

Como exemplo, usamos a seguinte função do Lambda como o back-end da API:

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

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

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

Comparando isso com a [configuração de integração personalizada do Lambda \(p. 240\)](#), 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.

Para configurar uma integração de proxy do Lambda usando a AWS CLI

1. Chame o comando `create-rest-api` para criar uma API:

```
aws apigateway create-rest-api --name 'HelloWorld (AWS CLI)' --region us-west-2
```

Observe o valor `id` da API resultante (`te6si5ach7`) na resposta:

```
{  
    "name": "HelloWorldProxy (AWS CLI)",  
    "id": "te6si5ach7",  
    "createdDate": 1508461860  
}
```

Você precisará do `id` da API durante toda esta seção.

2. Chame o comando `get-resources` para obter o `id` do recurso raiz:

```
aws apigateway get-resources --rest-api-id te6si5ach7 --region us-west-2
```

A resposta bem-sucedida é mostrada da seguinte forma:

```
{  
    "items": [  
        {  
            "path": "/",  
            "id": "krznpq9xpg"  
        }  
    ]  
}
```

Anote o valor `id` do recurso raiz (`krznpq9xpg`). Você precisará dele na próxima etapa e mais adiante.

3. Chame `create-resource` para criar um [Recurso](#) do API Gateway de `/greeting`:

```
aws apigateway create-resource --rest-api-id te6si5ach7 \  
    --region us-west-2 \  
    --parent-id krznpq9xpg \  
    --path-part {proxy+}
```

A resposta correta assemelha-se ao seguinte:

```
{  
    "path": "/{proxy+}",  
    "pathPart": "{proxy+}",  
    "id": "2jf6xt",  
    "parentId": "krznpq9xpg"  
}
```

Anote o valor `{proxy+}` do recurso `id` (`2jf6xt`). Você precisará dele para criar um método no recurso `/{proxy+}` na próxima etapa.

4. Chame `put-method` para criar uma solicitação de método ANY de ANY `/{proxy+}`:

```
aws apigateway put-method --rest-api-id te6si5ach7 \  
    --region us-west-2 \  
    --resource-id 2jf6xt \  
    --http-method ANY \  
    --authorization-type "NONE"
```

A resposta correta assemelha-se ao seguinte:

```
{  
    "apiKeyRequired": false,  
    "httpMethod": "ANY",  
    "authorizationType": "NONE"  
}
```

Esse método de API permite que o cliente receba ou envie saudações da função do Lambda no back-end.

5. Chame `put-integration` 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 \  
    --region us-west-2 \  
    --rest-api-id vaz7da96z6 \  
    --resource-id 2jf6xt \  
    --http-method ANY \  
    --type AWS_PROXY \  
    --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 \  
    --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 de ação do serviço 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 permissões do IAM, consulte [the section called “Modelo de permissões do API Gateway para invocar uma API” \(p. 378\)](#).

A saída bem-sucedida é semelhante seguinte:

```
{  
    "passthroughBehavior": "WHEN_NO_MATCH",
```

```
        "cacheKeyParameters": [],
        "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:1234567890:function:HelloWorld/invocations",
        "httpMethod": "POST",
        "cacheNamespace": "vvom7n",
        "credentials": "arn:aws:iam::1234567890:role/apigAwsProxyRole",
        "type": "AWS_PROXY"
}
```

Em vez de fornecer uma função do IAM para `credentials`, você pode chamar o comando [add-permission](#) para adicionar permissões baseadas em recurso. Isso é o que o console do API Gateway faz.

6. Chame `create-deployment` para implantar a API em um estágio `test`:

```
aws apigateway create-deployment --rest-api-id te6si5ach7 --stage-name test
```

7. 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' \
-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'
```

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 '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' \
-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 \
-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'
-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!
```

Formato de entrada de uma função Lambda para integração de proxy

Com a integração de proxy Lambda, o API Gateway mapeia toda a solicitação do cliente para o parâmetro `event` de entrada da função Lambda de back-end, da seguinte maneira:

```
{
  "resource": "Resource path",
  "path": "Path parameter",
```

```
"httpMethod": "Incoming request's method name"
"headers": {String containing incoming request headers}
"multiValueHeaders": {List of strings containing incoming request headers}
"queryStringParameters": {query string parameters }
"multiValueQueryStringParameters": {List of query string parameters}
"pathParameters": {path parameters}
"stageVariables": {Applicable stage variables}
"requestContext": {Request context, including authorizer-returned key-value pairs}
"body": "A JSON string of the request payload."
"isBase64Encoded": "A boolean flag to indicate if the applicable request payload is
Base64-encode"
}
```

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.

A seguinte solicitação POST mostra uma API implantada em `testStage` com a variável de estágio `stageVariableName=stageVariableValue`:

```
POST /testStage/hello/world?name=me HTTP/1.1
Host: gy415nuibc.execute-api.us-east-1.amazonaws.com
Content-Type: application/json
headerName: headerValue

{
    "a": 1
}
```

Essa solicitação acima produz a seguinte carga de resposta que contém a saída retornada da função do Lambda de back-end, em que `input` foi definido como o parâmetro `event` para a função do Lambda.

```
{
    "message": "Hello me!",
    "input": {
        "resource": "/{proxy+}",
        "path": "/hello/world",
        "httpMethod": "POST",
        "headers": {
            "Accept": "*/*",
            "Accept-Encoding": "gzip, deflate",
            "cache-control": "no-cache",
            "CloudFront-Forwarded-Proto": "https",
            "CloudFront-Is-Desktop-Viewer": "true",
            "CloudFront-Is-Mobile-Viewer": "false",
            "CloudFront-Is-SmartTV-Viewer": "false",
            "CloudFront-Is-Tablet-Viewer": "false",
            "CloudFront-Viewer-Country": "US",
            "Content-Type": "application/json",
            "headerName": "headerValue",
            "Host": "gy415nuibc.execute-api.us-east-1.amazonaws.com",
            "Postman-Token": "9f583ef0-ed83-4a38-aef3-eb9ce3f7a57f",
            "User-Agent": "PostmanRuntime/2.4.5",
            "Via": "CloudFront"
        },
        "stageVariables": {
            "stageVariableName": "stageVariableValue"
        }
    }
}
```

```
"Via": "1.1 d98420743a69852491bbdea73f7680bd.cloudfront.net (CloudFront)",  
"X-Amz-Cf-Id": "pn-PWIJc6thYnZm5P0NMgOUglL1DYtl0gdeJky8tqsg8iS_sgsKD1A==",  
"X-Forwarded-For": "54.240.196.186, 54.182.214.83",  
"X-Forwarded-Port": "443",  
"X-Forwarded-Proto": "https"  
},  
"multiValueHeaders":{  
    'Accept':[  
        "*/*"  
    ],  
    'Accept-Encoding':[  
        "gzip, deflate"  
    ],  
    'cache-control':[  
        "no-cache"  
    ],  
    'CloudFront-Forwarded-Proto':[  
        "https"  
    ],  
    'CloudFront-Is-Desktop-Viewer':[  
        "true"  
    ],  
    'CloudFront-Is-Mobile-Viewer':[  
        "false"  
    ],  
    'CloudFront-Is-SmartTV-Viewer':[  
        "false"  
    ],  
    'CloudFront-Is-Tablet-Viewer':[  
        "false"  
    ],  
    'CloudFront-Viewer-Country':[  
        "US"  
    ],  
    ':[  
        ""  
    ],  
    'Content-Type':[  
        "application/json"  
    ],  
    'headerName':[  
        "headerValue"  
    ],  
    'Host':[  
        "gy415nuibc.execute-api.us-east-1.amazonaws.com"  
    ],  
    'Postman-Token':[  
        "9f583ef0-ed83-4a38-aef3-eb9ce3f7a57f"  
    ],  
    'User-Agent':[  
        "PostmanRuntime/2.4.5"  
    ],  
    'Via':[  
        "1.1 d98420743a69852491bbdea73f7680bd.cloudfront.net (CloudFront)"  
    ],  
    'X-Amz-Cf-Id':[  
        "pn-PWIJc6thYnZm5P0NMgOUglL1DYtl0gdeJky8tqsg8iS_sgsKD1A=="  
    ],  
    'X-Forwarded-For':[  
        "54.240.196.186, 54.182.214.83"  
    ],  
    'X-Forwarded-Port':[  
        "443"  
    ],  
    'X-Forwarded-Proto':[  
        "https"  
    ]  
}
```

```
        ],
    },
    "queryStringParameters": {
        "name": "me",
        "multivalueName": "me"
    },
    "multiValueQueryStringParameters": [
        "name":[
            "me"
        ],
        "multivalueName":[
            "you",
            "me"
        ]
    },
    "pathParameters": {
        "proxy": "hello/world"
    },
    "stageVariables": {
        "stageVariableName": "stageVariableValue"
    },
    "requestContext": {
        "accountId": "12345678912",
        "resourceId": "roq9wj",
        "stage": "testStage",
        "requestId": "deef4878-7910-11e6-8f14-25afc3e9ae33",
        "identity": {
            "cognitoIdentityPoolId": null,
            "accountId": null,
            "cognitoIdentityId": null,
            "caller": null,
            "apiKey": null,
            "sourceIp": "192.168.196.186",
            "cognitoAuthenticationType": null,
            "cognitoAuthenticationProvider": null,
            "userArn": null,
            "userAgent": "PostmanRuntime/2.4.5",
            "user": null
        },
        "resourcePath": "/{proxy+}",
        "httpMethod": "POST",
        "apiId": "gy415nuibc"
    },
    "body": "{\r\n\t\"a\": 1\r\n}",
    "isBase64Encoded": false
}
```

Na entrada para a função Lambda do back-end, 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 \(p. 306\)](#), 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.

A tabela a seguir mostra um exemplo de um `requestContext` que é repassado para um endpoint de integração de proxy do Lambda quando o tipo de autorização é definido como AWS_IAM.

```
{  
    ...  
    "requestContext": {  
        "requestTime": "20/Feb/2018:22:48:57 +0000",  
        "path": "/test/",  
        "accountId": "123456789012",  
        "protocol": "HTTP/1.1",  
        "resourceId": "yx5mhem7ye",  
        "stage": "test",  
        "requestTimeEpoch": 1519166937665,  
        "requestId": "3c3ecbaa-1690-11e8-ae31-8f39f1d24af",  
        "identity": {  
            "cognitoIdentityPoolId": null,  
            "accountId": "123456789012",  
            "cognitoIdentityId": null,  
            "caller": "AIDAJ.....4HCKVJZG",  
            "sourceIp": "51.240.196.104",  
            "accessKey": "IAM_user_access_key",  
            "cognitoAuthenticationType": null,  
            "cognitoAuthenticationProvider": null,  
            "userArn": "arn:aws:iam::123456789012:user/alice",  
            "userAgent": "PostmanRuntime/7.1.1",  
            "user": "AIDAJ.....4HCKVJZG"  
        },  
        "resourcePath": "/",  
        "httpMethod": "GET",  
        "apiId": "qr2gd9cfmf"  
    },  
    ...  
}
```

Formato de saída de uma função Lambda para integração de proxy

Na integração de proxy Lambda, o API Gateway requer que a função Lambda de back-end retorne 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 para a 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 [Habilitar o suporte a binário usando o console do API Gateway \(p. 320\)](#).

Se a saída da função for de um formato diferente, o API Gateway retornará uma resposta de erro 502 `Bad Gateway`.

Para retornar uma resposta em uma função 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 `async` no Lambda no Node.js 8.10, 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 integrações personalizadas do Lambda no API Gateway

Para mostrar como configurar a integração personalizada do Lambda, criamos uma API do API Gateway para expor o método `GET /greeting?greeter={name}` para invocar uma função do Lambda. 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 `HelloWorld` à função.

Para referência, uma versão Node.js da função do Lambda é mostrada da seguinte forma:

```
exports.handler = function(event, context, callback) {
  var res ={
    "statusCode": 200,
    "headers": {
      "Content-Type": "*/*"
    }
  }
  if (event.queryStringParameters.greeter) {
    res.body = "Hello, " + event.queryStringParameters.greeter + "!";
  } else {
    res.body = "Hello, World!";
  }
  callback(null, res);
}
```

```
        }
    };
    if (event.greeter==null) {
        callback(new Error('Missing the required greeter parameter.'));
    } else if (event.greeter === "") {
        res.body = "Hello, World";
        callback(null, res);
    } else {
        res.body = "Hello, " + event.greeter +"!";
        callback(null, res);
    }
};
```

Você pode criar nele 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 back-end, configure a API.

Para configurar a integração personalizada do Lambda usando a AWS CLI

1. Chame o comando `create-rest-api` para criar uma API:

```
aws apigateway create-rest-api --name 'HelloWorld (AWS CLI)' --region us-west-2
```

Observe o valor `id` da API resultante (`te6si5ach7`) na resposta:

```
{
  "name": "HelloWorld (AWS CLI)",
  "id": "te6si5ach7",
  "createdDate": 1508461860
}
```

Você precisará do `id` da API durante toda esta seção.

2. Chame o comando `get-resources` para obter o `id` do recurso raiz:

```
aws apigateway get-resources --rest-api-id te6si5ach7 --region us-west-2
```

A resposta bem-sucedida é a seguinte:

```
{
  "items": [
    {
      "path": "/",
      "id": "krznpq9xpg"
    }
  ]
}
```

Anote o valor `id` do recurso raiz (`krznpq9xpg`). Você precisará dele na próxima etapa e mais adiante.

3. Chame `create-resource` para criar um [Recurso](#) do API Gateway de `/greeting`:

```
aws apigateway create-resource --rest-api-id te6si5ach7 \
--region us-west-2 \
```

```
--parent-id krznpq9xpg \
--path-part greeting
```

A resposta correta assemelha-se ao seguinte:

```
{
    "path": "/greeting",
    "pathPart": "greeting",
    "id": "2jf6xt",
    "parentId": "krznpq9xpg"
}
```

Anote o valor greeting do recurso id (2jf6xt). Você precisará dele para criar um método no recurso /greeting na próxima etapa.

4. Chame put-method para criar uma solicitação de método de API de GET /greeting? greeter={name}:

```
aws apigateway put-method --rest-api-id te6si5ach7 \
    --region us-west-2 \
    --resource-id 2jf6xt \
    --http-method GET \
    --authorization-type "NONE" \
    --request-parameters method.request.querystring.greeter=false
```

A resposta correta assemelha-se ao seguinte:

```
{
    "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 back-end. O parâmetro greeter é opcional, pois o back-end deve lidar com um chamador anônimo ou um chamador autoidentificado.

5. Chame put-method-response para configurar a resposta 200 OK para a solicitação de método de GET /greeting?greeter={name}:

```
aws apigateway put-method-response \
    --region us-west-2
    --rest-api-id te6si5ach7 \
    --resource-id 2jf6xt
    --http-method GET \
    --status-code 200
```

6. Chame put-integration 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 \
    --region us-west-2
    --rest-api-id vaz7da96z6 \
    --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 file://path/to/integration-request-template.json \
--credentials arn:aws:iam::123456789012:role/apigAwsProxyRole
```

Aqui, o valor do parâmetro `request-template`, `file://path/to/integration-request-template.json`, aponta para um arquivo JSON, chamado `integration-request-template.json` no diretório `path/to`, que contém um mapa de chave-valor como um objeto JSON. A chave é um tipo de mídia da carga útil da solicitação e o valor é um modelo de mapeamento para o corpo do tipo de conteúdo especificado. Neste exemplo, o arquivo JSON contém o seguinte objeto JSON:

```
{"application/json": "{\"greeter\": \"$input.params('greeter')\"}"}
```

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 na função do Lambda deve ser expressa no corpo. Você pode usar string JSON do mapa (por exemplo, `{"greeter": "'john'"}) como o valor de entrada request-template para o comando put-integration. No entanto, o uso da entrada de arquivo evita a difícil e, às vezes, impossível, colocação de aspas que é necessário para executar stringify para um objeto JSON.`

Important

Para integrações do Lambda, você deve usar o método HTTP de `POST` para a solicitação de integração, de acordo com a [especificação de ação do serviço Lambda para invocações de função](#). O parâmetro `uri` é o Nome de recurso da Amazon (ARN) da ação de invocação da função.

A saída bem-sucedida é semelhante ao seguinte:

```
{
  "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. Essa é a forma como o console do API Gateway adiciona essas permissões.

7. Chame `put-integration-response` para configurar a resposta de integração para transmitir a saída da função do Lambda para o cliente como a resposta de método `200 OK`.

```
aws apigateway put-integration-response \
--region us-west-2 \
--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 resposta correta deve ser semelhante ao seguinte:

```
{  
    "selectionPattern": "",  
    "statusCode": "200"  
}
```

8. Chame `create-deployment` para implantar a API em um estágio `test`:

```
aws apigateway create-deployment --rest-api-id te6si5ach7 --stage-name test
```

9. 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 Lambda do back-end

Na integração não proxy do Lambda (personalizada), a função do Lambda do back-end 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 back-end separado. Nesse caso, a função Lambda do back-end 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 Lambda](#). Isso é feito da seguinte forma:

Configurar a invocação assíncrona do Lambda no console do API Gateway

1. Em Integration Request (Solicitação de integração), adicione um cabeçalho `X-Amz-Invocation-Type`.
2. Em Method Request (Solicitação de método), adicione um cabeçalho `InvocationType` e o mapeie para o cabeçalho `x-Amz-Invocation-Type` na Integration Request (Solicitação de integração) com um valor estático de 'Event' ou com a expressão de mapeamento de cabeçalho de `method.request.header.InvocationType`. Para a expressão de mapeamento de cabeçalho, o cliente deve incluir o cabeçalho `InvocationType:Event` ao fazer uma solicitação para o método de API.

Lidar com 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"  
}
```

Nesta saída, statusCode normalmente é 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 statusCode especificado. 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

- [Lidar com erros padrão do Lambda no API Gateway \(p. 244\)](#)
- [Lidar com erros personalizados do Lambda no API Gateway \(p. 246\)](#)

Lidar com 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 Lambda do JavaScript (Node.js).

```
exports.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": [  
        "exports.handler (/var/task/index.js:3:14)"  
    ]  
}
```

Da mesma forma, considere a seguinte função do Lambda em 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. Em seguida, defina um padrão de expressão regular na propriedade `selectionPattern` do recurso `IntegrationResponse` associado ao código de status HTTP especificado. No console do API Gateway, esse `selectionPattern` é indicado como Lambda Error Regex (Regex de erro do Lambda) no editor de configuração Integration Response (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, para configurar uma nova expressão `selectionPattern`, usando a AWS CLI, chame o seguinte comando [put-integration-response](#):

```
aws apigateway put-integration-response --rest-api-id z0vprf0mdh --resource-id x3o5ih --
http-method GET --status-code 400 --selection-pattern "Invalid*" --region us-west-2
```

Certifique-se de configurar também o código de erro correspondente (400) na [resposta do método \(p. 213\)](#). 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.

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 Lambda Error Regex (Regex de erro Lambda) ao configurar ou atualizar uma resposta de integração de um código de status HTTP especificado.

Lidar com 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 Lambda do JavaScript (Node.js) retorna um erro personalizado:

```
exports.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 de corpo transforme a carga 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 mapeado para o cabeçalho `error_trace` como uma string JSON.

Configurar integrações HTTP 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 back-end, 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 \(p. 248\)](#)
- [Configurar integrações personalizadas HTTP no API Gateway \(p. 252\)](#)

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 back-end 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: Criar uma API com integração de proxy HTTP \(p. 54\)](#).

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 back-end:

```
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 back-end.

- 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 back-end:

```
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 back-end.

- 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 back-end:

```
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 back-end.

- 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 back-end:

```
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 back-end.

- 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 back-end:

```
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 back-end, 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 back-end:

```
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 back-end.

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 back-end 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 personalizada HTTP, você tem mais controle de quais dados transmitir entre um método de API e uma integração de API e como transmitir os dados. Isso é feito pelo mapeamento de dados.

Como parte da configuração da solicitação do método, você define 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 back-end. Em seguida, como parte da configuração da solicitação de integração, você define 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 back-end. Em seguida, como parte da configuração da resposta de integração, você define 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 a configuração de modelos de mapeamento, consulte [Configurar mapeamentos de dados. \(p. 270\)](#)

Configurar integrações privadas do API Gateway

A integração privada do API Gateway facilita a exposição de seus recursos de HTTP/HTTPS por trás de uma Amazon VPC para serem acessados pelos clientes de fora da VPC. Para estender o acesso aos recursos da sua VPC privada para além dos limites da VPC, você pode criar uma API com integração privada para acesso aberto ou acesso controlado. Você pode fazer isso usando as permissões do IAM, um autorizador do Lambda ou um grupo de usuários do Amazon Cognito.

A integração privada usa um recurso do API Gateway de `VpcLink` para encapsular as conexões entre os recursos do API Gateway e os da VPC visada. Como proprietário de um recurso da VPC, você é responsável por criar um Load balancer de rede na VPC e por adicionar um recurso da VPC como destino do listener de um Load balancer de rede. Como um desenvolvedor de API, para configurar uma API com a integração privada, você é responsável por criar um `VpcLink` direcionado ao Load balancer de rede especificado e por tratar o `VpcLink` como um endpoint real de integração.

Note

O Load balancer de rede e a API devem pertencer à mesma conta da AWS.

Com a integração privada do API Gateway, você pode habilitar o acesso aos recursos de HTTP/HTTPS dentro de uma VPC sem o conhecimento detalhado das configurações de redes privadas ou os dispositivos específicos de tecnologia.

Tópicos

- [Configurar um Load balancer de rede para Integrações privadas do API Gateway \(p. 253\)](#)
- [Conceder permissões para criar um link de VPC \(p. 253\)](#)
- [Configurar uma API do API Gateway com integrações privadas usando o console do API Gateway \(p. 254\)](#)
- [Configurar uma API do API Gateway com integrações privadas usando a AWS CLI \(p. 254\)](#)
- [Configurar a API com integrações privadas usando o OpenAPI \(p. 258\)](#)
- [Contas do API Gateway usadas para integrações privadas \(p. 259\)](#)

Configurar um Load balancer de rede para Integrações privadas do API Gateway

O procedimento a seguir descreve as etapas para configurar um Load balancer de rede 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: Criar uma API com integração privada” \(p. 89\)](#).

Note

O Load balancer de rede e a API devem pertencer à mesma conta da AWS.

Como criar um Load balancer de rede para integração privada usando o console do API Gateway

1. Faça login no console do Amazon EC2 em <https://console.aws.amazon.com/ec2/> e escolha uma região; por exemplo, us-east-1, na barra de navegação.
2. Configure um servidor web em uma instância do Amazon EC2. Para ver um exemplo de configuração, consulte [Instalação de um servidor web LAMP no Amazon Linux](#).
3. Crie um Load balancer de rede, registre a instância do EC2 com um grupo de destino e adicione o grupo de destino a um listener do Load balancer de rede. Para obter detalhes, siga as instruções em [Conceitos básicos de Load balancer de redes](#).

Depois que o Load balancer de rede for criado, anote o ARN dele. Ele será necessário para criar um link de VPC no API Gateway para integrar a API aos recursos da VPC por trás do Load balancer de rede.

Conceder permissões para criar um link de VPC

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 para criar e atualizar um **VpcLink**

1. Crie uma política do IAM semelhante a esta:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "ec2:CreateVpcEndpointServiceConfiguration",  
                "ec2:DeleteVpcEndpointServiceConfigurations",  
                "ec2:DescribeVpcEndpointServiceConfigurations",  
                "ec2:ModifyVpcEndpointServicePermissions"  
            ],  
            "Resource": "*"  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "elasticloadbalancing:DescribeLoadBalancers"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

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 o console do API Gateway

Para obter instruções de uso do console do API Gateway para configurar uma API com integração privada, consulte [TUTORIAL: Criar uma API com integração privada do API Gateway \(p. 89\)](#).

Configurar uma API do API Gateway com integrações privadas usando a AWS CLI

Antes de criar uma API com a integração privada, é necessário ter o recurso da VPC configurado e um Load balancer de rede criado e configurado com a origem da VPC como o destino. Se os requisitos não forem atendidos, siga [Configurar um Load balancer de rede para Integrações privadas do API Gateway \(p. 253\)](#) para instalar o recurso da VPC, criar um Load balancer de rede e definir o recurso da VPC como um destino do Load balancer de rede.

Note

O Load balancer de rede e a API devem pertencer à mesma conta da AWS.

Para que você possa criar e gerenciar um VpcLink, você também deve ter as permissões apropriadas configuradas. Para mais informações, consulte [Conceder permissões para criar um link de VPC \(p. 253\)](#).

Note

Você só precisa as permissões para criar um VpcLink na API. Você não precisa de permissões para usar o VpcLink.

Depois que o Load balancer de rede for criado, anote o ARN dele. Você precisará dele para criar um link de VPC para a integração privada.

Para configurar uma API com a integração privada usando a AWS CLI

1. Crie um VpcLink direcionado ao Load balancer de rede especificado.

Para esta discussão, suponhamos que o ARN do Load balancer de rede seja `arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/net/my-vpclink-test-nlb/1f8df693cd094a72`.

```
aws apigateway create-vpc-link \
--name my-test-vpc-link \
--target-arns arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/net/
my-vpclink-test-nlb/1f8df693cd094a72 \
--endpoint-url https://apigateway.us-east-1.amazonaws.com \
--region us-east-1
```

Se a configuração da AWS usa a região `us-east-1` como padrão, você pode ignorar os parâmetros `endpoint-url` e `region` da entrada anterior.

O comando anterior imediatamente retorna a seguinte resposta, confirmando o recebimento da solicitação e mostrando o status `PENDING` para o VpcLink que está sendo criado.

```
{
  "status": "PENDING",
  "targetArns": [
    "arn:aws:elasticloadbalancing:us-east-1:123456789012:loadbalancer/net/my-
vpclink-test-nlb/1f8df693cd094a72"
  ],
  "id": "gim7c3",
  "name": "my-test-vpc-link"
}
```

O API Gateway leva cerca de 2-4 minutos para concluir a criação do VpcLink. Quando a operação é concluída com êxito, o status é `AVAILABLE`. Você pode verificar isso usando o seguinte comando da CLI:

```
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 tentar criar um VpcLink com um Load balancer de rede que já esteja associado a um VPC endpoint, você obterá a seguinte mensagem na propriedade `statusMessage`:

```
"NLB is already associated with another VPC Endpoint Service"
```

Somente após a criação bem-sucedida do VpcLink estaremos prontos para criar a API e integrá-la ao recurso da VPC por meio do VpcLink.

Anote o valor do `id` do recém-criado VpcLink (`gim7c3` na saída anterior). Você precisará dele para configurar a integração privada.

2. Configure uma API criando um recurso de `RestApi` do API Gateway:

```
aws apigateway create-rest-api --name 'My VPC Link Test'
```

Nós abandonamos os parâmetros de entrada `endpoint-url` e `region` para usar a região padrão, conforme especificado na configuração da AWS.

Anote o valor do `RestApi` da `id` no resultado obtido. Neste exemplo, estamos supondo que ele é `6j4m3244we`. Você precisará desse valor para executar outras operações na API, incluindo a configuração de métodos e integrações.

Para fins de ilustração, criaremos uma API com apenas um método `GET` no recurso raiz (`/`) e integraremos o método ao `VpcLink`.

3. Configure o método `GET` `/`. Primeiro, obtenha o identificador do recurso raiz (`/`):

```
aws apigateway get-resources --rest-api-id 6j4m3244we
```

Na saída, anote o valor de `id` do caminho `/`. Neste exemplo, estamos supondo que ele seja `skpp60rab7`.

Configure a solicitação de método para o método da API `GET` `/`:

```
aws apigateway put-method \
--rest-api-id 6j4m3244we \
--resource-id skpp60rab7 \
--http-method GET \
--authorization-type "NONE"
```

Para usar as permissões do IAM, um autorizador do Lambda ou um grupo de usuários do Amazon Cognito para autenticar o chamador, defina `authorization-type` como `AWS_IAM`, `CUSTOM` ou `COGNITO_USER_POOLS`, respectivamente.

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. Vamos usar a integração de proxy aqui.

4. Configure a integração privada do tipo `HTTP_PROXY` e use o comando `put-integration` da seguinte forma:

```
aws apigateway put-integration \
--rest-api-id 6j4m3244we \
--resource-id skpp60rab7 \
--uri 'http://myApi.example.com' \
--http-method GET \
--type HTTP_PROXY \
--integration-http-method GET \
--connection-type VPC_LINK \
--connection-id gim7c3
```

Para uma integração privada, você deve definir `connection-type` como `VPC_LINK` e `connection-id` como o identificador do seu `VpcLink` ou como uma variável de estágio que faça referência ao ID do seu `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.

Se for bem-sucedido, o comando retornará a seguinte saída:

```
{
  "passthroughBehavior": "WHEN_NO_MATCH",
  "timeoutInMillis": 29000,
  "connectionId": "gim7c3",
  "uri": "http://myApi.example.com",
  "connectionType": "VPC_LINK",
  "httpMethod": "GET",
  "cacheNamespace": "skpp60rab7",
  "type": "HTTP_PROXY",
  "cacheKeyParameters": []}
```

}

Com uma variável de estágio, defina a propriedade `connectionId` ao criar a integração:

```
aws apigateway put-integration \
--rest-api-id 6j4m3244we \
--resource-id skpp60rab7 \
--uri 'http://myApi.example.com' \
--http-method GET \
--type HTTP_PROXY \
--integration-http-method GET \
--connection-type VPC_LINK \
--connection-id "\${stageVariables.vpcLinkId}"
```

Inclua aspas duplas na expressão da variável de estágio (`\${stageVariables.vpcLinkId}`) e insira um caractere de escape para o caractere `$`.

Como alternativa, você pode atualizar a integração para redefinir o valor de `connectionId` com uma variável de estágio:

```
aws apigateway update-integration \
--rest-api-id 6j4m3244we \
--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`.

O uso de uma variável de estágio para definir o valor do `connectionId` tem a vantagem de oferecer a mesma API integrada a diferentes `VpcLinks`, redefinindo o valor da variável de estágio. Isso é útil para mudar a API para outro link de VPC a fim de migrar para outro Load balancer de rede ou para outra VPC.

Como usamos a integração de proxy privada, a API agora está pronta para implantação e execução de testes. Com a integração não proxy, você também deve configurar a resposta do método e a resposta da integração da mesma forma que faria ao configurar uma [API com integrações personalizadas de HTTP](#) (p. 60).

5. Para testar a API, faça a sua implantação. Isso será necessário se você tiver usado a variável de estágio como um espaço reservado do ID do `VpcLink`. Para implantar a API com uma variável de estágio, use o comando `create-deployment` da seguinte forma:

```
aws apigateway create-deployment \
--rest-api-id 6j4m3244we \
--stage-name test \
--variables vpcLinkId=gim7c3
```

Para atualizar a variável de estágio com um ID diferente do `VpcLink` (por exemplo, `asf9d7`), use o comando `update-stage`:

```
aws apigateway update-stage \
--rest-api-id 6j4m3244we \
--stage-name test \
--patch-operations op=replace,path='/variables/vpcLinkId',value='asf9d7'
```

Para testar a API, chame-a usando o seguinte comando cURL:

```
curl -X GET https://6j4m3244we.beta.execute-api.us-east-1.amazonaws.com/test
```

Como alternativa, você pode digitar a URL de chamada da API em um navegador da web para visualizar o resultado.

Quando você codifica permanentemente a propriedade `connection-id` com o literal do ID do `VpcLink`, também pode chamar o `test-invoke-method` para testar a chamada da API antes de implantá-la.

Configurar a API com integrações privadas usando o 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 `VpcLink` ou como uma variável de estágio que faça referência ao ID de um `VpcLink`.
- 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 `VpcLink`. 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.vpcLinkId}`):

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "version": "2017-11-17T04:40:23Z",
    "title": "MyApiWithVpcLink"
  },
  "host": "p3wocvip9a.execute-api.us-west-2.amazonaws.com",
  "basePath": "/test",
  "schemes": [
    "https"
  ],
  "paths": {
    "/": {
      "get": {
        "produces": [
          "application/json"
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {
              "$ref": "#/definitions/Empty"
            }
          }
        }
      }
    },
    "x-amazon-apigateway-integration": {
      "httpMethod": "GET",
      "uri": "arn:aws:apigateway:us-west-2:lambda:path/2015-03-31/functions/arn:aws:lambda:us-west-2:123456789012:function:myFunction/invocations",
      "responses": {
        "200": {
          "statusCode": "200"
        }
      }
    }
  }
}
```

```
    "responses": {
        "default": {
            "statusCode": "200"
        }
    },
    "uri": "http://myApi.example.com",
    "passthroughBehavior": "when_no_match",
    "connectionType": "VPC_LINK",
    "connectionId": "${stageVariables.vpcLinkId}",
    "httpMethod": "GET",
    "type": "http_proxy"
}
}
},
"definitions": {
    "Empty": {
        "type": "object",
        "title": "Empty Schema"
    }
}
}
```

Contas do API Gateway usadas para integrações privadas

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-north-1	394634713161
ap-northeast-1	969236854626
ap-northeast-2	020402002396
ap-southeast-1	195145609632
ap-southeast-2	798376113853
ap-south-1	507069717855
ap-east-1	174803364771

Região	ID da conta
sa-east-1	287228555773
me-south-1	855739686837

Configurar integrações simuladas 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 back-end 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: Crie uma API REST importando um exemplo \(p. 45\)](#).

Como desenvolvedor de API, 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, nós descrevemos como usar o console do API Gateway para habilitar a integração simulada para um método da API.

Tópicos

- [Permitir integração simulada usando o console do API Gateway \(p. 261\)](#)

Permitir integração simulada usando o console do API Gateway

Você deve ter o método disponível no API Gateway. Siga as instruções em [TUTORIAL: Criar uma API com integração não proxy HTTP \(p. 59\)](#).

1. Escolha um recurso de API e crie um método. No painel Setup (Configuração) do método, escolha Mock (Simulação) no Integration type (Tipo de integração) e depois Save (Salvar).
2. Escolha Method Request (Solicitação de método) em Method Execution (Execução de método). Expanda URL Query String Parameters (Parâmetros de string de consulta de URL). Escolha Add query string (Adicionar string de consulta) para adicionar um parâmetro de consulta `scope`. Isso determina se o chamador é interno ou não.
3. Escolha Integration Request (Solicitação de integração) em Method Execution (Execução de método). Expanda Body Mapping Templates (Modelos de mapeamento do corpo). Escolha ou adicione um modelo de mapeamento application/json. Digite o seguinte no editor de modelo:

```
{  
    #if( $input.params('scope') == "internal" )  
        "statusCode": 200  
    #else  
        "statusCode": 500  
    #end  
}
```

Escolha Save (Salvar).

4. Escolha Integration Response (Resposta de integração) em Method Execution (Execução de método). Expanda a resposta 200 e, em seguida, a seção Body Mapping Templates (Modelos de mapeamento do corpo). Escolha ou adicione um modelo de mapeamento application/json e digite o seguinte modelo de mapeamento de corpo de resposta no editor de modelos.

```
{  
    "statusCode": 200,  
    "message": "Go ahead without me"  
}
```

Escolha Save (Salvar).

5. Role até Integration Response (Resposta de integração). Escolha Add integration response (Adicionar resposta de integração) para adicionar uma resposta 500. Digite `5\d{2}` em HTTP status regex (Regex de status HTTP). Expanda Body Mapping Templates (Modelos de mapeamento do corpo) e escolha Add mapping template (Adicionar modelo de mapeamento). Digite `application/json` para Content-Type (Tipo de conteúdo) e escolha o ícone de marca de seleção para salvar a configuração. No editor de modelo, digite o seguinte modelo de mapeamento de corpo de resposta de integração:

```
{  
    "statusCode": 500,  
    "message": "The invoked method is not supported on the API resource."  
}
```

Escolha Save (Salvar).

6. Escolha Test (Testar) em Method Execution (Execução de método). Faça o seguinte:
 - a. Digite `internal` em scope (escopo). Escolha Test. 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"}
```

6. Digite `public` em scope ou deixe em branco. Selecione 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.

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 back-end 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 sua API faz a intermediação entre um intercâmbio externo e a nuvem da AWS, você usa modelos de mapeamento VTL para a solicitação de integração ou a resposta de integração mapear a carga 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 `$context`, consulte [Variáveis \\$context para modelos de dados, autorizadores, modelos de mapeamento e registro de acesso em logs do CloudWatch](#) (p. 306). Para obter mais informações sobre `$stageVariables`, consulte [\\$stageVariables](#) (p. 315). Para obter mais informações sobre parâmetros de solicitação de método, consulte [Parâmetros de solicitação acessíveis por um modelo de mapeamento](#) (p. 301).

Tópicos

- [Respostas do gateway no API Gateway](#) (p. 263)
- [Tipos de respostas do gateway](#) (p. 264)
- [Configurar uma resposta do gateway usando o console do API Gateway](#) (p. 267)
- [Configurar uma resposta do gateway usando a API REST do API Gateway](#) (p. 269)
- [Configurar a personalização da resposta de gateway no OpenAPI](#) (p. 269)

Respostas do gateway 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 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` (p. 615).

Para permitir uma resposta de gateway, você configura uma resposta de gateway para um [tipo de resposta com suporte](#) (p. 264) em nível de API. Sempre que o API Gateway retorna uma resposta do tipo, os

mapeamentos de cabeçalho e os modelos de mapeamento de carga definidos na resposta de gateway são aplicados para retornar os resultados mapeados ao agente de chamada de API.

Na seção a seguir, mostramos como configurar respostas de gateway usando o console do API Gateway e a API REST do API Gateway.

Tipos de respostas do 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 DEFAULT_4XX.
API_CONFIGURATION_ERROR	500	A resposta do gateway para a configuração de API inválida, incluindo o endereço incorreto do endpoint enviado, a falha da decodificação em Base64 nos dados binários quando o suporte binário foi implementado ou o mapeamento da resposta de integração não podem 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 DEFAULT_5XX.
AUTHORIZER_CONFIGURATION_ERROR		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 DEFAULT_5XX.
AUTHORIZER_FAILURE	500	A resposta de gateway quando um autorizador personalizado ou do Amazon Cognito falhou ao autenticar o agente de chamada. Se o tipo de resposta não for especificado, essa resposta assumirá como padrão o tipo DEFAULT_5XX.
BAD_REQUEST_PARAMETERS	400	A resposta de gateway quando o parâmetro de solicitação não

Tipo de resposta do Gateway	Código de status de padrão	Descrição
		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 <code>DEFAULT_4XX</code> .
<code>BAD_REQUEST_BODY</code>	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 <code>DEFAULT_4XX</code> .
<code>DEFAULT_4XX</code>	Nulo	A resposta de gateway padrão para um tipo de resposta não especificado com o código de status de <code>4xx</code> . Alterar o código de status dessa resposta de gateway de retorno altera os códigos de status de todas as outras respostas <code>4xx</code> para o novo valor. Redefinir esse código de status para nulo reverte os códigos de status de todas as outras respostas <code>4xx</code> para seus valores originais.
<code>DEFAULT_5XX</code>	Nulo	A resposta de gateway padrão para um tipo de resposta não especificado com um código de status de <code>5xx</code> . Alterar o código de status dessa resposta de gateway de retorno altera os códigos de status de todas as outras respostas <code>5xx</code> para o novo valor. Redefinir esse código de status para nulo reverte os códigos de status de todas as outras respostas <code>5xx</code> para seus valores originais.
<code>EXPIRED_TOKEN</code>	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 <code>DEFAULT_4XX</code> .

Tipo de resposta do Gateway	Código de status de padrão	Descrição
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.
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.
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 o tipo DEFAULT_4XX.

Tipo de resposta do Gateway	Código de status de padrão	Descrição
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.
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 agente de 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.
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.

Configurar uma resposta do gateway usando o console do API Gateway

Para personalizar uma resposta de gateway usando o console do API Gateway

1. Faça login no console do API Gateway.
2. Escolha sua API existente ou crie uma nova.
3. Expanda a API no painel de navegação principal e escolha Respostas de gateway sob a API.
4. No painel Respostas de gateway, escolha um tipo de resposta. Nesta demonstração, usamos Token de autenticação ausente (403) como um 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 sua 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 Add Header (Adicionar cabeçalho) em Response Headers (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 mapeamento de corpo, deixe application/json para Tipo de conteúdo e digite o seguinte modelo de mapeamento no editor Modelo de mapeamento de corpo:

```
{  
    "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 do gateway.

8. Escolha Salvar.
9. Implantar a API em um estágio novo ou existente.
10. Teste-a chamando o seguinte comando CURL, supondo 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 será retornado para acionar a 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: a2be05a4-2ee5-11e7-bbf2-df131ec50ae6
Access-Control-Allow-Origin: a.b.c
x-amzn-ErrorType: MissingAuthenticationTokenException
header-1: static
x-request-query: 1
x-request-path: 5
X-Cache: Error from cloudfront
Via: 1.1 441811a054e8d055b893175754efd0c3.cloudfront.net (CloudFront)
X-Amz-Cf-Id: nNDR-fX4csbRoAgtQJ16u0rTDz9FZWT-Mk93KgoxnfzDlTUh3flmzA==

{
    "message": "Missing Authentication Token",
    "type": MISSING_AUTHENTICATION_TOKEN,
    "statusCode": '404',
    "stage": custErr,
    "resourcePath": /pets/{petId},
    "stageVariables.a": a
}
```

O exemplo anterior pressupõe que o back-end da API seja [Pet Store](#) e que a API possua uma variável de estágio, a, definida.

Configurar uma resposta do 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 inteira de [GatewayResponse](#), chame a ação `gatewayresponse:put`, especificando um `responseType` desejado no parâmetro do caminho da URL e fornecendo na carga da solicitação os mapeamentos `statusCode`, `responseParameters` e `responseTemplates`.
2. Para atualizar parte de uma instância de [GatewayResponse](#), chame a ação `gatewayresponse:update`, especificando um `responseType` desejado no parâmetro do caminho da URL e fornecendo na carga da solicitação as propriedades [GatewayResponse](#) individuais 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            }\n        "
        }
    }
}
```

```
\",\n      \"resourcePath\": \"$context.resourcePath\",\\n      \"stageVariables.a\":\n      \"$stageVariables.a\",\\n      \"statusCode\": \"404\"\n    }\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.

Configurar mapeamentos de dados de solicitações e respostas do API Gateway

Tópicos

- [Configurar mapeamentos de dados de solicitação e resposta usando o console do API Gateway \(p. 270\)](#)
- [Criar modelos e modelos de mapeamento para mapeamentos de solicitação e resposta \(p. 273\)](#)
- [Referência de mapeamento de dados de solicitações e respostas de API do Amazon API Gateway \(p. 301\)](#)
- [Referência de variáveis de registro de acesso em logs e modelo de mapeamento do API Gateway \(p. 305\)](#)

Configurar mapeamentos de dados de solicitação e resposta usando o console do API Gateway

Para usar o console do API Gateway para definir a solicitação/resposta de integração da API, siga estas instruções.

Note

Estas instruções supõem que você já concluiu as etapas em [Configurar uma solicitação de integração de API usando o console do API Gateway \(p. 223\)](#).

1. Com o método selecionado no painel Resources (Recursos), no painel Method Execution (Execução de método), escolha Integration Request (Solicitação de resposta).
2. Para um proxy HTTP ou um proxy de serviço da AWS, para associar um parâmetro de caminho, um parâmetro de string de consulta ou um parâmetro de cabeçalho definido na solicitação de integração a um parâmetro de caminho, um parâmetro de string de consulta ou um parâmetro de cabeçalho correspondente na solicitação de método do proxy HTTP ou do proxy de serviço da AWS, faça o seguinte:
 - a. Escolha a seta ao lado de URL Path Parameters (Parâmetros de caminho de URL), URL Query String Parameters (Parâmetros de string de consulta de URL), ou HTTP Headers (Cabeçalhos HTTP) respectivamente, e escolha Add path (Adicionar caminho), Add query string (Adicionar string de consulta), ou Add header (Adicionar cabeçalho), respectivamente.
 - b. Para Name (Nome), digite o nome do parâmetro de caminho, do parâmetro da string de consulta ou do parâmetro de cabeçalho no proxy HTTP ou proxy de serviço da AWS.
 - c. Para Mapped from (Mapeado de), digite o valor de mapeamento para o parâmetro de caminho, o parâmetro de string de consulta ou o parâmetro de cabeçalho. Use um dos seguintes formatos:
 - **method.request.path.parameter-name** para um parâmetro de caminho denominado **parameter-name**, como definido na página Method Request (Solicitação de método).
 - **method.request.querystring.parameter-name** para um parâmetro de string de consulta denominado **parameter-name**, como definido na página Method Request (Solicitação de método).

- `method.request.multivaluequerystring.parameter-name` para um parâmetro de string de consulta de vários valores denominado `parameter-name`, como definido na página Method Request (Solicitação de método).
- `method.request.header.parameter-name` para um parâmetro de cabeçalho denominado `parameter-name`, como definido na página Method Request (Solicitação de método).

Como alternativa, você pode definir um valor de string literal (delimitado por um par de aspas simples) para um cabeçalho de integração.

- `method.request.multivalueheader.parameter-name` para um parâmetro de cabeçalho de vários valores denominado `parameter-name`, como definido na página Method Request (Solicitação de método).

- d. Escolha Criar. (Para excluir um parâmetro de caminho, um parâmetro de string de consulta ou um parâmetro de cabeçalho, escolha Cancel (Cancelar) ou Remove (Remover) ao lado do parâmetro que você deseja excluir.)

3. Na área Body Mapping Templates (Modelos de mapeamento do corpo), escolha uma opção para Request body passthrough (Passagem do corpo da solicitação) de forma a configurar como o corpo da solicitação de método de um tipo de conteúdo não mapeado será transmitido através da solicitação de integração sem transformação para a função do Lambda, o proxy HTTP ou o proxy de serviço da AWS. Existem três opções:

- Escolha When no template matches the request Content-Type header (Quando nenhum modelo corresponde ao cabeçalho Content-Type) 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, conforme definido na próxima etapa.

Note

Ao chamar a API do API Gateway, escolha essa opção configurando `WHEN_NO_MATCH` como o valor da propriedade `passthroughBehavior` no recurso [Integração](#).

- Escolha When there are no templates defined (recommended) (Quando não há modelos definidos (recomendado)) se quiser que o corpo da solicitação de método transmita a solicitação de integração ao 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.

Note

Ao chamar a API do API Gateway, escolha essa opção configurando `WHEN_NO_TEMPLATE` como o valor da propriedade `passthroughBehavior` no recurso [Integração](#).

- Escolha Never (Nunca) se não quiser que a solicitação de método seja transmitida quando seu tipo de conteúdo não corresponder a nenhum tipo de conteúdo associado aos modelos de mapeamento definidos na solicitação de integração ou quando nenhum modelo de mapeamento estiver definido na solicitação de integração. A solicitação de método de um tipo de conteúdo não mapeado será rejeitada com uma resposta HTTP 415 Tipo de mídia sem suporte.

Note

Ao chamar a API do API Gateway, escolha essa opção configurando `NEVER` como o valor da propriedade `passthroughBehavior` no recurso [Integração](#).

Para obter mais informações sobre os comportamentos de passagem direta de integração, consulte [Comportamentos de passagem direta de integração \(p. 304\)](#).

4. Para definir um modelo de mapeamento para uma solicitação recebida, escolha Add mapping template (Adicionar modelo de mapeamento) sob Content-Type (Tipo de conteúdo). Digite um tipo

de conteúdo (por exemplo, `application/json`) na caixa de texto de entrada e depois escolha o ícone de marca de seleção para salvar a entrada. Em seguida, digite o modelo de mapeamento manualmente ou escolha Generate template (Gerar modelo) para criar um partir de um modelo de modelo. Para obter mais informações, consulte [Criar modelos e modelos de mapeamento para mapeamentos de solicitação e resposta \(p. 273\)](#).

5. Você pode mapear uma resposta de integração no back-end para uma resposta de método da API retornada ao aplicativo de chamada. Isso inclui retornar ao cliente os cabeçalhos de resposta selecionados nos cabeçalhos disponíveis no back-end, transformando o formato de dados da carga da resposta do back-end em um formato especificado pela API. Você pode especificar tal mapeamento configurando Method Response (Resposta de método) e Integration Response (Resposta de integração) na página Method Execution (Execução de método).
 - a. No painel Method Execution (Execução de método), escolha Integration Response (Resposta de integração). Escolha a seta ao lado de 200 para especificar configurações para um código de resposta 200 HTTP do método, ou escolha Add integration response (Adicionar resposta de integração) para especificar configurações para qualquer outro código de status de resposta HTTP do método.
 - b. Para Lambda error regex (Regex de erro do Lambda) (para uma função do Lambda) ou HTTP status regex (Regex de status HTTP) (para um proxy HTTP ou proxy de serviço da AWS), digite uma expressão regular para especificar quais strings de erro da função do Lambda (para uma função do Lambda) ou quais códigos de status de resposta HTTP (para um proxy HTTP ou proxy de serviço da AWS) são mapeados para esse mapeamento de saída. Por exemplo, para mapear todos os códigos de status de resposta HTTP 2xx de um proxy HTTP para esse mapeamento de saída, digite "`2\\d{2}`" em HTTP status regex (Regex de status HTTP). Para retornar uma mensagem de erro contendo "Solicitação inválida" de uma função do Lambda para uma resposta 400 Bad Request, digite "`.*Invalid request.*`" como a expressão Lambda error regex (Regex de erro do Lambda). Por outro lado retornar 400 Bad Request para todas as mensagens de erro não mapeadas do Lambda, digite "`(\\n|.)*`" em Lambda error regex (Regex de erro do Lambda). Essa última expressão regular pode ser usada para a resposta de erro padrão de uma API.

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.

Os padrões de erro são comparados à string inteira da propriedade `errorMessage` na resposta do Lambda, que é preenchida por `callback(errorMessage)` em Node.js ou por `throw new MyException(errorMessage)` em Java. Além disso, caracteres escapados têm o escape cancelado antes que a expressão regular seja aplicada.

Se você usar "+" como o padrão de seleção para respostas de filtro, lembre-se de que ele pode não corresponder a uma resposta que contém um caractere de nova linha ("\\n").

- c. Se ativado, para Method response status (Status de resposta de método), escolha o código de status de resposta HTTP definido na página Method Response (Resposta de método).
- d. Para Header Mappings (Mapeamentos de cabeçalho), para cada cabeçalho que você definiu para o código de status de resposta HTTP na página Method Response (Resposta de método), especifique um valor de mapeamento escolhendo Edit (Editar). Em Mapping value (Valor de mapeamento), use um dos seguintes formatos:
 - `integration.response.multivalueheaders.header-name` onde `header-name` é o nome de um cabeçalho de resposta de vários valores do back-end.

Por exemplo, para retornar o cabeçalho Date da resposta do back-end como um cabeçalho Timestamp da resposta de um método de API, a coluna Response header (Cabeçalho da resposta) conterá uma entrada Timestamp, e o Mapping value (Valor de mapeamento) associado deve ser definido como `integration.response.multivalueheaders.Date`.

- **integration.response.header.header-name** onde **header-name** é o nome de um cabeçalho de resposta de valor único do back-end.

Por exemplo, para retornar o cabeçalho Date da resposta do back-end como um cabeçalho Timestamp da resposta de um método de API, a coluna Response header (Cabeçalho da resposta) conterá uma entrada Timestamp, e o Mapping value (Valor de mapeamento) associado deve ser definido como integration.response.header.Date.

- e. Na área Template Mappings (Mapeamentos de modelo), próximo a Content type (Tipo de conteúdo), escolha Add (Adicionar). Na caixa Content type (Tipo de conteúdo), digite o tipo de conteúdo dos dados que serão transmitidos da função do Lambda, do proxy HTTP ou do proxy de serviço da AWS ao método. Escolha Update.
- f. Selecione Output passthrough (Passagem de saída) se você deseja o método para receber, mas não modificar, os dados da função do Lambda, proxy HTTP ou proxy de serviço da AWS.
- g. Se Output passthrough (Passagem de saída) estiver desmarcado, para Output mapping (Mapeamento de saída), especifique o modelo de mapeamento de saída que você deseja que a função do Lambda, o proxy HTTP ou o proxy de serviço da AWS utilize para enviar dados ao método. Você pode digitar o modelo de mapeamento manualmente ou escolher um modelo de Generate template from model (Gerar modelo a partir do modelo).
- h. Escolha Salvar.

Criar modelos e modelos de mapeamento para mapeamentos de solicitação e resposta

No Gateway da API, a solicitação de método de uma API pode usar uma carga em um formato diferente da carga da solicitação de integração correspondente, conforme exigido no back-end. De maneira semelhante, o back-end 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 modelos de mapeamento para mapear a carga de uma solicitação de método para a solicitação de integração correspondente ou de uma resposta de integração para a resposta de método correspondente.

Um modelo de mapeamento é um script expresso em [Velocity Template Language \(VTL\)](#) e aplicado à carga usando [expressões JSONPath](#).

A carga pode ter um modelo de dados de acordo com o [esquema JSON rascunho 4](#). Você deve definir o modelo para que o API Gateway gere um SDK ou habilite a validação básica de solicitações para a sua API. Você não precisa definir um modelo para criar um modelo de mapeamento. No entanto, um modelo pode ajudá-lo a criar um modelo, pois o API Gateway gerará um blueprint de modelo com base em um modelo fornecido.

A seção explica como mapear a carga de solicitação e resposta da API usando modelos e modelos de mapeamento.

Tópicos

- [Modelos \(p. 274\)](#)
- [Modelos de mapeamento \(p. 277\)](#)
- [Tarefas para modelos e modelos de mapeamento \(p. 279\)](#)
- [Criar um modelo no API Gateway \(p. 279\)](#)
- [Exibir uma lista de modelos no API Gateway \(p. 280\)](#)
- [Como usar um modelo de mapeamento para substituir parâmetros de solicitação e resposta e códigos de status de API \(p. 280\)](#)
- [Excluir um modelo no API Gateway \(p. 285\)](#)
- [Exemplo de fotos \(modelos e modelos de mapeamento do API Gateway\) \(p. 285\)](#)

- [Exemplo de matéria jornalística \(modelos e modelos de mapeamento do API Gateway\) \(p. 289\)](#)
- [Exemplo de fatura de vendas \(modelos e modelos de mapeamento do API Gateway\) \(p. 292\)](#)
- [Exemplo de registro de funcionário \(modelos e modelos de mapeamento do API Gateway\) \(p. 296\)](#)

Modelos

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 seguinte objeto JSON descreve dados de amostra que descrevem o estoque de frutas ou vegetais no departamento de produtos de um supermercado:

Suponha que tenhamos uma API para o gerenciamento do estoque de frutas e vegetais no departamento de produto de um supermercado. Quando um gerente consulta o back-end para o estoque atual, o servidor retorna a seguinte carga de resposta:

```
{  
  "department": "produce",  
  "categories": [  
    "fruit",  
    "vegetables"  
,  
    "bins": [  
      {  
        "category": "fruit",  
        "type": "apples",  
        "price": 1.99,  
        "unit": "pound",  
        "quantity": 232  
      },  
      {  
        "category": "fruit",  
        "type": "bananas",  
        "price": 0.19,  
        "unit": "each",  
        "quantity": 112  
      },  
      {  
        "category": "vegetables",  
        "type": "carrots",  
        "price": 1.29,  
        "unit": "bag",  
        "quantity": 57  
      }  
    ]  
}
```

O objeto JSON tem três propriedades

- A propriedade `department` tem um valor de string (`produce`).
- A propriedade `categories` é uma matriz de duas strings: `fruit` e `vegetables`.
- A propriedade `bins` é uma matriz de objetos, cada um com as propriedades de valor de string ou número de `category`, `type`, `price`, `unit` e `quantity`.

Podemos usar o seguinte Esquema JSON para definir o modelo para os dados acima:

```
{  
  "$schema": "http://json-schema.org/draft-04/schema#",
```

```
"title": "GroceryStoreInputModel",
"type": "object",
"properties": {
    "department": { "type": "string" },
    "categories": {
        "type": "array",
        "items": { "type": "string" }
    },
    "bins": {
        "type": "array",
        "items": {
            "type": "object",
            "properties": {
                "category": { "type": "string" },
                "type": { "type": "string" },
                "price": { "type": "number" },
                "unit": { "type": "string" },
                "quantity": { "type": "integer" }
            }
        }
    }
}
```

No modelo de exemplo anterior:

- O objeto `$schema` representa um identificador de versão do Esquema JSON válido. Neste exemplo, ele faz referência ao Esquema JSON, rascunho v4.
- O objeto `title` é um identificador humanamente legível para o modelo. Neste exemplo, é `GroceryStoreInputModel`.
- A construção de nível superior, ou raiz, nos dados JSON é um objeto.
- O objeto raiz nos dados JSON contém as propriedades `department`, `categories` e `bins`.
- A propriedade `department` é um objeto de string dos dados JSON.
- A propriedade `categories` é uma matriz nos dados JSON. A matriz contém valores de string nos dados JSON.
- A propriedade `bins` é uma matriz nos dados JSON. A matriz contém objetos nos dados JSON. Cada um desses objetos nos dados JSON contém uma string `category`, uma string `type`, um número `price`, uma string `unit` e um número inteiro `quantity` (um número sem uma fração ou parte exponencial).

Como alternativa, você poderia incluir parte desse esquema, por exemplo, a definição de item da matriz `bins`, em uma seção separada do mesmo arquivo e usar o primitivo `$ref` para fazer referência a essa definição reutilizável em outras partes do esquema. Usando `$ref`, o arquivo de definição de modelo acima pode ser expresso da seguinte maneira:

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "GroceryStoreInputModel",
    "type": "object",
    "properties": {
        "department": { "type": "string" },
        "categories": {
            "type": "array",
            "items": { "type": "string" }
        },
        "bins": {
            "type": "array",
            "items": {
                "$ref": "#/definitions/Bin"
            }
        }
}
```

```
        },
    "definitions": {
        "Bin" : {
            "type": "object",
            "properties": {
                "category": { "type": "string" },
                "type": { "type": "string" },
                "price": { "type": "number" },
                "unit": { "type": "string" },
                "quantity": { "type": "integer" }
            }
        }
    }
}
```

A seção `definitions` contém a definição de esquema do item `Bin` referenciado na matriz `bins` com `"$ref": "#/definitions/Bin"`. Usar definições reutilizáveis dessa forma facilita sua definição de modelo.

Além disso, você também pode consultar outro esquema de modelo definido em um arquivo de modelo externo, definindo a URL do modelo como o valor da propriedade `$ref": "https://apigateway.amazonaws.com/restapis/{restapi_id}/models/{model_name}"`. Por exemplo, suponha que você tenha o seguinte modelo desenvolvido denominado `Bin` criado em uma API com um identificador `fugvwdxtri`:

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "GroceryStoreInputModel",
    "type": "object",
    "properties": {
        "Bin" : {
            "type": "object",
            "properties": {
                "category": { "type": "string" },
                "type": { "type": "string" },
                "price": { "type": "number" },
                "unit": { "type": "string" },
                "quantity": { "type": "integer" }
            }
        }
    }
}
```

Você pode fazer referência a ele em `GroceryStoreInputModel`, a partir da mesma API, conforme mostrado a seguir:

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "GroceryStoreInputModel",
    "type": "object",
    "properties": {
        "department": { "type": "string" },
        "categories": {
            "type": "array",
            "items": { "type": "string" }
        },
        "bins": {
            "type": "array",
            "items": {
                "$ref": "https://apigateway.amazonaws.com/restapis/fugvwdxtri/models/Bin2"
            }
        }
    }
}
```

```
    }
}
```

Os modelos de referência e referenciados devem ser da mesma API.

Os exemplos não usam recursos avançados de Esquema JSON, como especificar itens necessários; comprimentos de string mínimos e máximos permitidos, valores numéricos e comprimentos de itens da matriz, expressões regulares e muito mais. Para obter mais informações, consulte [Apresentação do JSON e esquema JSON rascunho 4](#).

Para formatos de dados JSON mais complexos e seus modelos, consulte os seguintes exemplos:

- [Modelo de entrada \(Exemplo de fotos\) \(p. 286\)](#) e [Modelo de saída \(Exemplo de fotos\) \(p. 288\)](#) no [Exemplo de fotos \(p. 285\)](#)
- [Modelo de entrada \(Exemplo de matéria jornalística\) \(p. 289\)](#) e [Modelo de saída \(Exemplo de matéria jornalística\) \(p. 291\)](#) no [Exemplo de matéria jornalística \(p. 289\)](#)
- [Modelo de entrada \(Exemplo de fatura de vendas\) \(p. 293\)](#) e [Modelo de saída \(Exemplo de fatura de vendas\) \(p. 295\)](#) no [Exemplo de fatura de vendas \(p. 292\)](#)
- [Modelo de entrada \(exemplo de registro de funcionário\) \(p. 297\)](#) e [Modelo de saída \(exemplo de registro de funcionário\) \(p. 299\)](#) no [Exemplo de registro de funcionário \(p. 296\)](#)

Para testar modelos no API Gateway, siga as instruções em [Mapear carga da resposta \(p. 77\)](#), especificamente [Etapa 1: criar modelos \(p. 79\)](#).

Modelos de mapeamento

Quando o back-end retorna os resultados da consulta (mostrados na seção [Modelos \(p. 274\)](#)), o gerente do departamento de produtos pode estar interessado em lê-los da seguinte maneira:

```
{
  "choices": [
    {
      "kind": "apples",
      "suggestedPrice": "1.99 per pound",
      "available": 232
    },
    {
      "kind": "bananas",
      "suggestedPrice": "0.19 per each",
      "available": 112
    },
    {
      "kind": "carrots",
      "suggestedPrice": "1.29 per bag",
      "available": 57
    }
  ]
}
```

Para permitir isso, precisamos fornecer ao API Gateway um modelo de mapeamento para converter os dados a partir do formato de back-end. O seguinte modelo de mapeamento fará exatamente isso.

```
#set($inputRoot = $input.path('$'))
{
  "choices": [
    #foreach($elem in $inputRoot.bins)
```

```
{  
    "kind": "$elem.type",  
    "suggestedPrice": "$elem.price per $elem.unit",  
    "available": $elem.quantity  
}#if($foreach.hasNext),#end  
  
#end  
]  
}
```

Vejamos agora alguns detalhes do modelo de mapeamento de saída anterior:

- A variável `$inputRoot` representa o objeto raiz nos dados JSON originais da seção anterior. As variáveis em um modelo de mapeamento de saída são mapeadas para os dados JSON originais, e não para o esquema de dados JSON transformado desejado.
- A matriz `choices` no modelo de mapeamento de saída é mapeada da matriz `bins` com o objeto raiz nos dados JSON originais (`$inputRoot.bins`).
- No modelo de mapeamento de saída, cada um dos objetos na matriz `choices` (representada por `$elem`) é mapeado dos objetos correspondentes na matriz `bins` dentro do objeto raiz nos dados JSON originais.
- No modelo de mapeamento de saída, para cada um dos objetos no objeto `choices`, os valores dos objetos `kind` e `available` (representados por `$elem.type` e `$elem.quantity`) são mapeados a partir dos valores correspondentes dos objetos `type` e `value` em cada um dos objetos na matriz `bins` dos dados JSON originais, respectivamente.
- No modelo de mapeamento de saída, para cada um dos objetos no objeto `choices`, o valor do objeto `suggestedPrice` é uma concatenação do valor correspondente dos objetos `price` e `unit` em cada um dos objetos nos dados JSON originais, respectivamente, com cada valor separado pela palavra `per`.

Para obter mais informações sobre o VTL (Velocity Template Language), consulte o documento [Apache Velocity - VTL Reference](#). Para obter mais informações sobre o JSONPath, consulte [JSONPath - XPath para JSON](#).

O modelo de mapeamento pressupõe que os dados subjacentes sejam de um objeto JSON. Ele não exige que um modelo seja definido para os dados. Como um desenvolvedor de API, você conhece os formatos de dados no front-end e no back-end. Esse conhecimento pode orientá-lo para definir os mapeamentos necessários sem ambiguidade.

Para ter um SDK gerado para a API, os dados acima serão retornados como um objeto específico de uma linguagem. 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 criará esse UDT se você fornecer a ele um modelo de dados. Para o exemplo de resposta do método acima, você pode definir o seguinte modelo de carga na resposta de integração:

```
{  
    "$schema": "http://json-schema.org/draft-04/schema#",  
    "title": "GroceryStoreOutputModel",  
    "type": "object",  
    "properties": {  
        "choices": {  
            "type": "array",  
            "items": {  
                "type": "object",  
                "properties": {  
                    "kind": { "type": "string" },  
                    "suggestedPrice": { "type": "string" },  
                    "available": { "type": "integer" }  
                }  
            }  
        }  
    }  
}
```

```
    }  
}
```

Nesse modelo, o esquema JSON é expresso da seguinte maneira:

- O objeto `$schema` representa um identificador de versão do Esquema JSON válido. Neste exemplo, ele faz referência ao Esquema JSON, rascunho v4.
- O objeto `title` é um identificador humanamente legível para o modelo. Neste exemplo, é `GroceryStoreOutputModel`.
- A construção de nível superior, ou raiz, nos dados JSON é um objeto.
- O objeto raiz nos dados JSON contém uma matriz de objetos.
- Cada objeto na matriz de objetos contém uma string `kind`, uma string `suggestedPrice` e um número inteiro `available` (um número sem uma fração ou parte exponencial).

Com esse modelo, você pode chamar um SDK para recuperar os valores de propriedades `kind`, `suggestedPrice` e `available`, lendo as propriedades `GroceryStoreOutputModel[i].kind`, `GroceryStoreOutputModel[i].suggestedPrice` e `GroceryStoreOutputModel[i].available`, respectivamente. Se nenhum modelo for fornecido, o API Gateway usará o modelo Vazio para criar um UDT padrão. Nesse caso, você não poderá ler essas propriedades usando um SDK de tipo forte.

Para explorar modelos de mapeamento mais complexos, consulte os exemplos a seguir:

- [Modelo de mapeamento de entrada \(Exemplo de fotos\) \(p. 287\)](#) e [Modelo de mapeamento de saída \(Exemplo de fotos\) \(p. 288\)](#) no [Exemplo de fotos \(p. 285\)](#)
- [Modelo de mapeamento de entrada \(Exemplo de matéria jornalística\) \(p. 290\)](#) e [Modelo de mapeamento de saída \(Exemplo de matéria jornalística\) \(p. 291\)](#) no [Exemplo de matéria jornalística \(p. 289\)](#)
- [Modelo de mapeamento de entrada \(Exemplo de fatura de vendas\) \(p. 294\)](#) e [Modelo de mapeamento de saída \(Exemplo de fatura de vendas\) \(p. 295\)](#) no [Exemplo de fatura de vendas \(p. 292\)](#)
- [Modelo de mapeamento de entrada \(Exemplo de registro de funcionário\) \(p. 298\)](#) e [Modelo de mapeamento de saída \(Exemplo de registro de funcionário\) \(p. 300\)](#) no [Exemplo de registro de funcionário \(p. 296\)](#)

Para testar modelos de mapeamento no API Gateway, siga as instruções em [Mapear carga da resposta \(p. 77\)](#), especificamente [Etapa 5: configurar e testar os métodos \(p. 84\)](#).

Tarefas para modelos e modelos de mapeamento

Para conhecer as coisas adicionais que você pode fazer com modelos e modelos de mapeamento, consulte o seguinte:

- [Criar um modelo \(p. 279\)](#)
- [Exibir uma lista de modelos \(p. 280\)](#)
- [Excluir um modelo \(p. 285\)](#)

Criar um modelo no API Gateway

Usar o console do API Gateway para criar um modelo para uma API.

Tópicos

- [Pré-requisitos \(p. 280\)](#)

- Criar um modelo com o console do API Gateway (p. 280)

Pré-requisitos

- Você deve ter uma API disponível no API Gateway. Siga as instruções em [Criação de uma API REST no Amazon API Gateway \(p. 182\)](#).

Criar um modelo com o console do API Gateway

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Na caixa que contém o nome da API na qual você deseja criar o modelo, selecione Models (Modelos).
3. Escolha Criar.
4. Para Model Name, digite um nome para o modelo.
5. Em Content Type, digite o tipo de conteúdo do modelo (por exemplo, `application/json` para o JSON).
6. (Opcional) Para Model description, digite uma descrição para o modelo.
7. Para o Model schema, digite o esquema do modelo. Para obter mais informações sobre esquemas de modelo, consulte [Criar modelos e modelos de mapeamento para mapeamentos de solicitação e resposta \(p. 273\)](#).
8. Escolha Create model (Criar modelo).

Exibir uma lista de modelos no API Gateway

Usar o console do API Gateway para exibir uma lista de modelos.

Tópicos

- [Pré-requisitos \(p. 280\)](#)
- [Exibir uma lista de modelos com o console do API Gateway \(p. 280\)](#)

Pré-requisitos

- Você deve ter pelo menos um modelo no API Gateway. Siga as instruções em [Criar um modelo \(p. 279\)](#).

Exibir uma lista de modelos com o console do API Gateway

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Na caixa que contém o nome da API, escolha Modelos.

Como usar um modelo de mapeamento para substituir parâmetros de solicitação e resposta e códigos de status de API

Os [modelos de mapeamento de parâmetro e código de resposta \(p. 273\)](#) padrão do API Gateway que permitem que você mapeie parâmetros individualmente e mapeie uma família de códigos de status de resposta de integração (correspondidos por uma expressão regular) para um único código de status de resposta. As substituições por meio de modelo de mapeamento oferecem a flexibilidade de executar mapeamentos de parâmetro "muitos para um"; substitua os parâmetros após a aplicação de mapeamentos padrão do API Gateway; mapeie condicionalmente os parâmetros com base no conteúdo do corpo ou outros valores de parâmetro; crie novos parâmetros programaticamente e instantaneamente; e substitua

códigos de status retornados pelo endpoint de integração. Qualquer tipo de parâmetro de solicitação, cabeçalho de resposta ou código de status de resposta pode ser substituído.

A seguir encontram-se exemplos de uso de substituições por meio de modelo de mapeamento:

- Criar um novo cabeçalho (ou substituir um cabeçalho existente) como uma concatenação de dois parâmetros
- Substituir o código de resposta para um código de êxito ou falha com base no conteúdo do corpo
- Remapar condicionalmente um parâmetro com base em seu conteúdo ou no conteúdo de algum outro parâmetro
- Iterar o conteúdo de um corpo json e remapar pares de chave/valor para cabeçalhos ou strings de consulta

Para criar uma substituição por meio de modelo de mapeamento, use uma ou mais das seguintes [\\$context variáveis \(p. 306\)](#) em um [modelo de mapeamento \(p. 273\)](#):

Modelo de mapeamento do corpo da solicitação	Modelo de mapeamento do corpo da resposta
<code>\$context.requestOverride.header.<i>header_name</i></code>	<code>\$context.responseOverride.header.<i>header_name</i></code>
<code>\$context.requestOverride.path.<i>path_name</i></code>	<code>\$context.responseOverride.status</code>
<code>\$context.requestOverride.querystring.<i>querystring_name</i></code>	

Note

Não é possível usar substituições por meio de modelo de mapeamento com endpoints de integração de proxy, os quais não têm mapeamentos de dados. Para mais informações sobre integração, consulte [Escolher um tipo de integração da API do API Gateway \(p. 221\)](#).

Important

As substituições são feitas no final. Uma substituição só pode ser aplicada a um parâmetro por vez. A tentativa de substituir o mesmo parâmetro várias vezes gerará respostas 5XX do Amazon API Gateway. 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. Observe que o modelo é aplicado somente depois que todo o modelo é analisado. Consulte [Tutorial: como substituir parâmetros e cabeçalhos de solicitação de uma API com o console do API Gateway \(p. 283\)](#).

Os tutoriais a seguir mostram como criar e testar uma substituição por meio de modelo de mapeamento no console do API Gateway. Esses tutoriais usam a [API de exemplo PetStore \(p. 45\)](#) como ponto de partida. Os dois tutoriais supõem que você já tenha criado a [API de exemplo PetStore \(p. 45\)](#).

Tópicos

- [Tutorial: Como substituir o código de status de resposta de uma API com o console do API Gateway \(p. 282\)](#)
- [Tutorial: como substituir parâmetros e cabeçalhos de solicitação de uma API com o console do API Gateway \(p. 283\)](#)
- [Exemplos: Como substituir parâmetros e cabeçalhos de solicitação de uma API com a CLI do API Gateway \(p. 284\)](#)
- [Exemplo: Como substituir parâmetros e cabeçalhos de solicitação de uma API usando o SDK for JavaScript \(p. 284\)](#)

Tutorial: Como substituir o código de status de resposta de uma API com o console do API Gateway

Para recuperar um animal de estimação usando a API de exemplo PetStore, use 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.

Neste tutorial, você vai substituir esse código de resposta do método GET criando um modelo de mapeamento que mapeia `$context.responseOverride.status` para 400 quando é detectada uma condição de erro.

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Em APIs, escolha a API PetStore.
3. Na coluna Resources (Recursos), escolha o método GET em `/{{petId}}`.
4. Na caixa Client (Cliente), escolha Test (Testar).
5. Digite `-1` para `{petId}` e escolha Test (Testar).

Nos resultados, você verá dois fatores:

Primeiro, Response Body (Corpo da resposta) indica um erro fora do intervalo:

```
{
  "errors": [
    {
      "key": "GetPetRequest.petId",
      "message": "The value is out of range."
    }
  ]
}
```

Segundo, a última linha na caixa Logs termina com: Method completed with status: 200.

6. Volte para Method Execution (Execução de método). Escolha Integration Response (Resposta de integração) e depois selecione a seta ao lado de 200.
7. Na seção Mapping Templates (Modelos de mapeamento), escolha Add mapping template (Adicionar modelo de mapeamento).
8. Em Content Type (Tipo de conteúdo), digite **application/json** e, em seguida, escolha o ícone de marca de seleção para salvar a escolha.
9. Copie o seguinte código para a área do modelo:

```
#set($inputRoot = $input.path('$'))
$input.json("$")
#if($inputRoot.toString().contains("error"))
#set($context.responseOverride.status = 400)
#end
```

10. Escolha Salvar.
11. Volte para Method Execution (Execução de método)
12. Na caixa Client (Cliente), escolha Test (Testar).
13. Digite `-1` para `{petId}` e escolha Test (Testar).

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 na caixa Logs agora termina com: Method completed with status: 400.

Tutorial: como substituir parâmetros e cabeçalhos de solicitação de uma API com o console do API Gateway

Neste tutorial, você vai substituir o código do cabeçalho da solicitação do método GET criando um modelo de mapeamento que é mapeia `$context.requestOverride.header.header_name` para um novo cabeçalho que associa dois outros cabeçalhos.

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Em APIs, escolha a API PetStore.
3. Na coluna Resources (Recursos), escolha o método GET em /pets.
4. Escolha Method Request (Solicitação de método).
5. Crie um parâmetro como se segue:
 - a. Expanda HTTP Request Headers (Cabeçalhos de solicitação HTTP).
 - b. Escolha Add header (Adicionar cabeçalho).
 - c. Em Name (Nome), digite **header1**.
 - d. Escolha o ícone de marca de seleção para salvar sua escolha.

Repita o processo para criar um segundo cabeçalho chamado header2.

6. Volte para Method Execution (Execução de método)
7. Escolha Integration Request (Solicitação de integração).
8. Expanda HTTP Headers (Cabeçalhos HTTP). Você verá os dois cabeçalhos criados, `header1` e `header2`, junto com seus mapeamentos padrão (em Mapped from (Mapeado de)).
9. Expanda Mapping Templates (Modelos de mapeamento).
10. Escolha Add mapping template (Adicionar modelo de mapeamento).
11. Em Content Type (Tipo de conteúdo), digite **application/json** e, em seguida, escolha o ícone de marca de seleção para salvar a escolha.
12. Uma pop-up será exibida com a informação Note: This template can map headers and body. (Observação: Este modelo pode mapear cabeçalhos e corpo.).

Escolha Yes, secure this integration (Sim, garantir essa integração).

13. Copie o seguinte código para a área do modelo:

```
#set($header1Override = "foo")
#set($header3Value = "$input.params('header1')$input.params('header2')")
$input.json("$")
#set($context.requestOverride.header.header3 = $header3Value)
#set($context.requestOverride.header.header1 = $header1Override)
#set($context.requestOverride.header.multivalueheader=[$header1Override,
$header3Value])
```

14. Escolha Salvar.
15. Volte para Method Execution (Execução de método)

16. Na caixa Client (Cliente), escolha Test (Testar).
17. Em Headers (Cabeçalhos) para {pets}, copie o seguinte código:

```
header1:header1Val  
header2:header2Val
```

18. Escolha Test.

Em Logs, você deve ver uma entrada que inclui este texto:

```
Endpoint request headers: {header3=header1Valheader2Val,  
header2=header2Val, header1=foo, x-amzn-apigateway-api-id=<api-id>,  
Accept=application/json, multivalueheader=foo,header1Valheader2Val}
```

Exemplos: Como substituir parâmetros e cabeçalhos de solicitação de uma API com a CLI do API Gateway

O seguinte exemplo de CLI mostra como usar o comando `put-integration` para substituir um código de resposta:

```
aws apigateway put-integration --rest-api-id <API_ID> --resource-id <PATH_TO_RESOURCE_ID>  
--http-method <METHOD>  
--type <INTEGRATION_TYPE> --request-templates <REQUEST_TEMPLATE_MAP>
```

onde `<REQUEST_TEMPLATE_MAP>` é um mapa do tipo de conteúdo para uma string do modelo a ser aplicado. A estrutura do mapa é como a seguir:

```
Content_type1=template_string,Content_type2=template_string
```

ou, na sintaxe JSON:

```
{"content_type1": "template_string"  
...}
```

O seguinte exemplo mostra como usar o comando `put-integration-response` para substituir o código de resposta de uma API:

```
aws apigateway put-integration-response --rest-api-id <API_ID> --resource-  
id <PATH_TO_RESOURCE_ID> --http-method <METHOD>  
--status-code <STATUS_CODE> --response-templates <RESPONSE_TEMPLATE_MAP>
```

onde `<RESPONSE_TEMPLATE_MAP>` tem o mesmo formato que `<REQUEST_TEMPLATE_MAP>` acima.

Exemplo: Como substituir parâmetros e cabeçalhos de solicitação de uma API usando o SDK for JavaScript

O seguinte exemplo mostra como usar o comando `put-integration` para substituir um código de resposta:

Solicitação:

```
var params = {  
  httpMethod: 'STRING_VALUE', /* required */
```

```
resourceId: 'STRING_VALUE', /* required */  
restApiId: 'STRING_VALUE', /* required */  
type: HTTP | AWS | MOCK | HTTP_PROXY | AWS_PROXY, /* required */  
requestTemplates: {  
    '<Content_type>': 'TEMPLATE_STRING',  
    /* '<String>': ... */  
},  
};  
apigateway.putIntegration(params, function(err, data) {  
    if (err) console.log(err, err.stack); // an error occurred  
    else     console.log(data);           // successful response  
});
```

Resposta:

```
var params = {  
    httpMethod: 'STRING_VALUE', /* required */  
    resourceId: 'STRING_VALUE', /* required */  
    restApiId: 'STRING_VALUE', /* required */  
    statusCode: 'STRING_VALUE', /* required */  
    responseTemplates: {  
        '<Content_type>': 'TEMPLATE_STRING',  
        /* '<String>': ... */  
    },  
};  
apigateway.putIntegrationResponse(params, function(err, data) {  
    if (err) console.log(err, err.stack); // an error occurred  
    else     console.log(data);           // successful response  
});
```

Excluir um modelo no API Gateway

Usar o console do API Gateway para excluir um modelo.

Warning

A exclusão de um modelo pode fazer com que parte da API correspondente, ou toda ela, se torne inutilizável pelos agentes de chamadas de API. A exclusão de um modelo não pode ser desfeita.

Excluir um modelo com o console do API Gateway

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Na caixa que contém o nome da API para o modelo, escolha Models.
3. No painel Models, escolha o modelo que você deseja excluir e escolha Delete Model.
4. Quando solicitado, escolha Delete.

Exemplo de fotos (modelos e modelos de mapeamento do API Gateway)

As seções a seguir fornecem exemplos de modelos e modelos de mapeamento que podem ser usados para uma amostra de API de fotos no API Gateway. Para obter mais informações sobre modelos e modelos de mapeamento no API Gateway, consulte [Criar modelos e modelos de mapeamento para mapeamentos de solicitação e resposta \(p. 273\)](#).

Tópicos

- [Dados originais \(Exemplo de fotos\) \(p. 286\)](#)
- [Modelo de entrada \(Exemplo de fotos\) \(p. 286\)](#)
- [Modelo de mapeamento de entrada \(Exemplo de fotos\) \(p. 287\)](#)

- [Dados transformados \(Exemplo de fotos\) \(p. 287\)](#)
- [Modelo de saída \(Exemplo de fotos\) \(p. 288\)](#)
- [Modelo de mapeamento de saída \(Exemplo de fotos\) \(p. 288\)](#)

Dados originais (Exemplo de fotos)

Os dados a seguir são os dados JSON originais para o exemplo de fotos:

```
{  
  "photos": {  
    "page": 1,  
    "pages": "1234",  
    "perpage": 100,  
    "total": "123398",  
    "photo": [  
      {  
        "id": "12345678901",  
        "owner": "23456789@A12",  
        "secret": "abc123d456",  
        "server": "1234",  
        "farm": 1,  
        "title": "Sample photo 1",  
        "ispublic": 1,  
        "isfriend": 0,  
        "isfamily": 0  
      },  
      {  
        "id": "23456789012",  
        "owner": "34567890@B23",  
        "secret": "bcd234e567",  
        "server": "2345",  
        "farm": 2,  
        "title": "Sample photo 2",  
        "ispublic": 1,  
        "isfriend": 0,  
        "isfamily": 0  
      }  
    ]  
  }  
}
```

Modelo de entrada (Exemplo de fotos)

O modelo de entrada a seguir corresponde aos dados JSON originais para o exemplo de fotos:

```
{  
  "$schema": "http://json-schema.org/draft-04/schema#",  
  "title": "PhotosInputModel",  
  "type": "object",  
  "properties": {  
    "photos": {  
      "type": "object",  
      "properties": {  
        "page": { "type": "integer" },  
        "pages": { "type": "string" },  
        "perpage": { "type": "integer" },  
        "total": { "type": "string" },  
        "photo": {  
          "type": "array",  
          "items": {  
            "type": "object",  
            "properties": {  
              "id": "12345678901",  
              "owner": "23456789@A12",  
              "secret": "abc123d456",  
              "server": "1234",  
              "farm": 1,  
              "title": "Sample photo 1",  
              "ispublic": 1,  
              "isfriend": 0,  
              "isfamily": 0  
            }  
          }  
        }  
      }  
    }  
  }  
}
```

```
        "properties": {
            "id": { "type": "string" },
            "owner": { "type": "string" },
            "secret": { "type": "string" },
            "server": { "type": "string" },
            "farm": { "type": "integer" },
            "title": { "type": "string" },
            "ispublic": { "type": "integer" },
            "isfriend": { "type": "integer" },
            "isfamily": { "type": "integer" }
        }
    }
}
}
}
```

Modelo de mapeamento de entrada (Exemplo de fotos)

O modelo de mapeamento de entrada a seguir corresponde aos dados JSON originais para o exemplo de fotos:

```
#set($inputRoot = $input.path('$'))
{
    "photos": {
        "page": $inputRoot.photos.page,
        "pages": "$inputRoot.photos.pages",
        "perpage": $inputRoot.photos.perpage,
        "total": "$inputRoot.photos.total",
        "photo": [
#foreach($elem in $inputRoot.photos.photo)
            {
                "id": "$elem.id",
                "owner": "$elem.owner",
                "secret": "$elem.secret",
                "server": "$elem.server",
                "farm": $elem.farm,
                "title": "$elem.title",
                "ispublic": $elem.ispublic,
                "isfriend": $elem.isfriend,
                "isfamily": $elem.isfamily
            }#if($foreach.hasNext),#end
#end
        ]
    }
}
```

Dados transformados (Exemplo de fotos)

O exemplo a seguir mostra como os dados JSON do exemplo de fotos original podem ser transformados para saída:

```
{
    "photos": [
        {
            "id": "12345678901",
            "owner": "23456789@A12",
            "title": "Sample photo 1",
            "ispublic": 1,
            "isfriend": 0,
```

```
        "isfamily": 0
    },
    {
        "id": "23456789012",
        "owner": "34567890@B23",
        "title": "Sample photo 2",
        "ispublic": 1,
        "isfriend": 0,
        "isfamily": 0
    }
]
```

Modelo de saída (Exemplo de fotos)

Veja a seguir o modelo de saída que corresponde ao formato de dados JSON transformado:

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "PhotosOutputModel",
    "type": "object",
    "properties": {
        "photos": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "id": { "type": "string" },
                    "owner": { "type": "string" },
                    "title": { "type": "string" },
                    "ispublic": { "type": "integer" },
                    "isfriend": { "type": "integer" },
                    "isfamily": { "type": "integer" }
                }
            }
        }
    }
}
```

Modelo de mapeamento de saída (Exemplo de fotos)

Veja a seguir o modelo de mapeamento de saída que corresponde ao formato de dados JSON transformado. Aqui, as variáveis de modelo se baseiam no formato de dados JSON original, não transformado:

```
#set($inputRoot = $input.path('$'))
{
    "photos": [
        #foreach($elem in $inputRoot.photos.photo)
        {
            "id": "$elem.id",
            "owner": "$elem.owner",
            "title": "$elem.title",
            "ispublic": $elem.ispublic,
            "isfriend": $elem.isfriend,
            "isfamily": $elem.isfamily
        }#if($foreach.hasNext),#end
    ]
}
```

Exemplo de matéria jornalística (modelos e modelos de mapeamento do API Gateway)

As seções a seguir fornecem exemplos de modelos e modelos de mapeamento que podem ser usados para uma amostra de API de matéria jornalística no API Gateway. Para obter mais informações sobre modelos e modelos de mapeamento no API Gateway, consulte [Criar modelos e modelos de mapeamento para mapeamentos de solicitação e resposta \(p. 273\)](#).

Tópicos

- [Dados originais \(Exemplo de matéria jornalística\) \(p. 289\)](#)
- [Modelo de entrada \(Exemplo de matéria jornalística\) \(p. 289\)](#)
- [Modelo de mapeamento de entrada \(Exemplo de matéria jornalística\) \(p. 290\)](#)
- [Dados transformados \(Exemplo de matéria jornalística\) \(p. 291\)](#)
- [Modelo de saída \(Exemplo de matéria jornalística\) \(p. 291\)](#)
- [Modelo de mapeamento de saída \(Exemplo de matéria jornalística\) \(p. 291\)](#)

Dados originais (Exemplo de matéria jornalística)

Os dados a seguir são os dados JSON originais para o exemplo de matéria jornalística:

```
{  
    "count": 1,  
    "items": [  
        {  
            "last_updated_date": "2015-04-24",  
            "expire_date": "2016-04-25",  
            "author_first_name": "John",  
            "description": "Sample Description",  
            "creation_date": "2015-04-20",  
            "title": "Sample Title",  
            "allow_comment": "1",  
            "author": {  
                "last_name": "Doe",  
                "email": "johndoe@example.com",  
                "first_name": "John"  
            },  
            "body": "Sample Body",  
            "publish_date": "2015-04-25",  
            "version": "1",  
            "author_last_name": "Doe",  
            "parent_id": 2345678901,  
            "article_url": "http://www.example.com/articles/3456789012"  
        }  
    ],  
    "version": 1  
}
```

Modelo de entrada (Exemplo de matéria jornalística)

O modelo de entrada a seguir corresponde aos dados JSON originais para o exemplo de matéria jornalística:

```
{  
    "$schema": "http://json-schema.org/draft-04/schema#",  
    "title": "NewsArticleInputModel",  
    "type": "object",  
    "properties": {  
        "count": { "type": "integer" },  
        "items": {  
            "type": "array",  
            "items": {  
                "type": "object",  
                "properties": {  
                    "last_updated_date": { "type": "string", "format": "date" },  
                    "expire_date": { "type": "string", "format": "date" },  
                    "author_first_name": { "type": "string" },  
                    "description": { "type": "string" },  
                    "creation_date": { "type": "string", "format": "date" },  
                    "title": { "type": "string" },  
                    "allow_comment": { "type": "string" },  
                    "author": {  
                        "type": "object",  
                        "properties": {  
                            "last_name": { "type": "string" },  
                            "email": { "type": "string" },  
                            "first_name": { "type": "string" }  
                        }  
                    },  
                    "body": { "type": "string" },  
                    "publish_date": { "type": "string", "format": "date" },  
                    "version": { "type": "string" },  
                    "author_last_name": { "type": "string" },  
                    "parent_id": { "type": "integer" },  
                    "article_url": { "type": "string" }  
                }  
            }  
        }  
    }  
}
```

```
"type": "array",
"items": {
    "type": "object",
    "properties": {
        "last_updated_date": { "type": "string" },
        "expire_date": { "type": "string" },
        "author_first_name": { "type": "string" },
        "description": { "type": "string" },
        "creation_date": { "type": "string" },
        "title": { "type": "string" },
        "allow_comment": { "type": "string" },
        "author": {
            "type": "object",
            "properties": {
                "last_name": { "type": "string" },
                "email": { "type": "string" },
                "first_name": { "type": "string" }
            }
        },
        "body": { "type": "string" },
        "publish_date": { "type": "string" },
        "version": { "type": "string" },
        "author_last_name": { "type": "string" },
        "parent_id": { "type": "integer" },
        "article_url": { "type": "string" }
    }
},
"version": { "type": "integer" }
}
```

Modelo de mapeamento de entrada (Exemplo de matéria jornalística)

O modelo de mapeamento de entrada a seguir corresponde aos dados JSON originais para o exemplo de matéria jornalística:

```
#set($inputRoot = $input.path('$'))
{
    "count": $inputRoot.count,
    "items": [
#foreach($elem in $inputRoot.items)
    {
        "last_updated_date": "$elem.last_updated_date",
        "expire_date": "$elem.expire_date",
        "author_first_name": "$elem.author_first_name",
        "description": "$elem.description",
        "creation_date": "$elem.creation_date",
        "title": "$elem.title",
        "allow_comment": "$elem.allow_comment",
        "author": {
            "last_name": "$elem.author.last_name",
            "email": "$elem.author.email",
            "first_name": "$elem.author.first_name"
        },
        "body": "$elem.body",
        "publish_date": "$elem.publish_date",
        "version": "$elem.version",
        "author_last_name": "$elem.author_last_name",
        "parent_id": $elem.parent_id,
        "article_url": "$elem.article_url"
    }#if($foreach.hasNext),#end
    #end
}
```

```
    ],
    "version": $inputRoot.version
}
```

Dados transformados (Exemplo de matéria jornalística)

O exemplo a seguir mostra como os dados JSON do exemplo de matéria jornalística original podem ser transformados para saída:

```
{
  "count": 1,
  "items": [
    {
      "creation_date": "2015-04-20",
      "title": "Sample Title",
      "author": "John Doe",
      "body": "Sample Body",
      "publish_date": "2015-04-25",
      "article_url": "http://www.example.com/articles/3456789012"
    }
  ],
  "version": 1
}
```

Modelo de saída (Exemplo de matéria jornalística)

Veja a seguir o modelo de saída que corresponde ao formato de dados JSON transformado:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "NewsArticleOutputModel",
  "type": "object",
  "properties": {
    "count": { "type": "integer" },
    "items": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "creation_date": { "type": "string" },
          "title": { "type": "string" },
          "author": { "type": "string" },
          "body": { "type": "string" },
          "publish_date": { "type": "string" },
          "article_url": { "type": "string" }
        }
      }
    },
    "version": { "type": "integer" }
  }
}
```

Modelo de mapeamento de saída (Exemplo de matéria jornalística)

Veja a seguir o modelo de mapeamento de saída que corresponde ao formato de dados JSON transformado. Aqui, as variáveis de modelo se baseiam no formato de dados JSON original, não transformado:

```
#set($inputRoot = $input.path('$'))
{
  "count": $inputRoot.count,
  "items": [
```

```
#foreach($elem in $inputRoot.items)
{
    "creation_date": "$elem.creation_date",
    "title": "$elem.title",
    "author": "$elem.author.first_name $elem.author.last_name",
    "body": "$elem.body",
    "publish_date": "$elem.publish_date",
    "article_url": "$elem.article_url"
}#if($foreach.hasNext),#end

#end
],
"version": $inputRoot.version
}
```

Exemplo de fatura de vendas (modelos e modelos de mapeamento do API Gateway)

As seções a seguir fornecem exemplos de modelos e modelos de mapeamento que podem ser usados para uma amostra de API de fatura de vendas no API Gateway. Para obter mais informações sobre modelos e modelos de mapeamento no API Gateway, consulte [Criar modelos e modelos de mapeamento para mapeamentos de solicitação e resposta \(p. 273\)](#).

Tópicos

- [Dados originais \(Exemplo de fatura de vendas\) \(p. 292\)](#)
- [Modelo de entrada \(Exemplo de fatura de vendas\) \(p. 293\)](#)
- [Modelo de mapeamento de entrada \(Exemplo de fatura de vendas\) \(p. 294\)](#)
- [Dados transformados \(Exemplo de fatura de vendas\) \(p. 295\)](#)
- [Modelo de saída \(Exemplo de fatura de vendas\) \(p. 295\)](#)
- [Modelo de mapeamento de saída \(Exemplo de fatura de vendas\) \(p. 295\)](#)

Dados originais (Exemplo de fatura de vendas)

Os dados a seguir são os dados JSON originais para o exemplo de fatura de vendas:

```
{
    "DueDate": "2013-02-15",
    "Balance": 1990.19,
    "DocNumber": "SAMP001",
    "Status": "Payable",
    "Line": [
        {
            "Description": "Sample Expense",
            "Amount": 500,
            "DetailType": "ExpenseDetail",
            "ExpenseDetail": {
                "Customer": {
                    "value": "ABC123",
                    "name": "Sample Customer"
                },
                "Ref": {
                    "value": "DEF234",
                    "name": "Sample Construction"
                },
                "Account": {
                    "value": "EFG345",
                    "name": "Fuel"
                }
            },
            "LineStatus": "Billable"
```

```
        }
    ],
    "Vendor": {
        "value": "GHI456",
        "name": "Sample Bank"
    },
    "APRef": {
        "value": "HIJ567",
        "name": "Accounts Payable"
    },
    "TotalAmt": 1990.19
}
```

Modelo de entrada (Exemplo de fatura de vendas)

O modelo de entrada a seguir corresponde aos dados JSON originais para o exemplo de fatura de vendas:

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "InvoiceInputModel",
    "type": "object",
    "properties": {
        "DueDate": { "type": "string" },
        "Balance": { "type": "number" },
        "DocNumber": { "type": "string" },
        "Status": { "type": "string" },
        "Line": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "Description": { "type": "string" },
                    "Amount": { "type": "integer" },
                    "DetailType": { "type": "string" },
                    "ExpenseDetail": {
                        "type": "object",
                        "properties": {
                            "Customer": {
                                "type": "object",
                                "properties": {
                                    "value": { "type": "string" },
                                    "name": { "type": "string" }
                                }
                            }
                        }
                    },
                    "Ref": {
                        "type": "object",
                        "properties": {
                            "value": { "type": "string" },
                            "name": { "type": "string" }
                        }
                    }
                }
            }
        },
        "Account": {
            "type": "object",
            "properties": {
                "value": { "type": "string" },
                "name": { "type": "string" }
            }
        },
        "LineStatus": { "type": "string" }
    }
}
```

```
"Vendor": {
    "type": "object",
    "properties": {
        "value": { "type": "string" },
        "name": { "type": "string" }
    }
},
"APRef": {
    "type": "object",
    "properties": {
        "value": { "type": "string" },
        "name": { "type": "string" }
    }
},
"TotalAmt": { "type": "number" }
}
```

Modelo de mapeamento de entrada (Exemplo de fatura de vendas)

O modelo de mapeamento de entrada a seguir corresponde aos dados JSON originais para o exemplo de fatura de vendas:

```
#set($inputRoot = $input.path('$'))
{
    "DueDate": "$inputRoot.DueDate",
    "Balance": $inputRoot.Balance,
    "DocNumber": "$inputRoot.DocNumber",
    "Status": "$inputRoot.Status",
    "Line": [
        #foreach($elem in $inputRoot.Line)
        {
            "Description": "$elem.Description",
            "Amount": $elem.Amount,
            "DetailType": "$elem.DetailType",
            "ExpenseDetail": {
                "Customer": {
                    "value": "$elem.ExpenseDetail.Customer.value",
                    "name": "$elem.ExpenseDetail.Customer.name"
                },
                "Ref": {
                    "value": "$elem.ExpenseDetail.Ref.value",
                    "name": "$elem.ExpenseDetail.Ref.name"
                },
                "Account": {
                    "value": "$elem.ExpenseDetail.Account.value",
                    "name": "$elem.ExpenseDetail.Account.name"
                },
                "LineStatus": "$elem.ExpenseDetail.LineStatus"
            }
        }#if($foreach.hasNext),#end
    ],
    "Vendor": {
        "value": "$inputRoot.Vendor.value",
        "name": "$inputRoot.Vendor.name"
    },
    "APRef": {
        "value": "$inputRoot.APRef.value",
        "name": "$inputRoot.APRef.name"
    },
    "TotalAmt": $inputRoot.TotalAmt
}
```

Dados transformados (Exemplo de fatura de vendas)

O exemplo a seguir mostra como os dados JSON do exemplo de fatura de vendas original podem ser transformados para saída:

```
{  
    "DueDate": "2013-02-15",  
    "Balance": 1990.19,  
    "DocNumber": "SAMP001",  
    "Status": "Payable",  
    "Line": [  
        {  
            "Description": "Sample Expense",  
            "Amount": 500,  
            "DetailType": "ExpenseDetail",  
            "Customer": "ABC123 (Sample Customer)",  
            "Ref": "DEF234 (Sample Construction)",  
            "Account": "EFG345 (Fuel)",  
            "LineStatus": "Billable"  
        }  
    ],  
    "TotalAmt": 1990.19  
}
```

Modelo de saída (Exemplo de fatura de vendas)

Veja a seguir o modelo de saída que corresponde ao formato de dados JSON transformado:

```
{  
    "$schema": "http://json-schema.org/draft-04/schema#",  
    "title": "InvoiceOutputModel",  
    "type": "object",  
    "properties": {  
        "DueDate": { "type": "string" },  
        "Balance": { "type": "number" },  
        "DocNumber": { "type": "string" },  
        "Status": { "type": "string" },  
        "Line": {  
            "type": "array",  
            "items": {  
                "type": "object",  
                "properties": {  
                    "Description": { "type": "string" },  
                    "Amount": { "type": "integer" },  
                    "DetailType": { "type": "string" },  
                    "Customer": { "type": "string" },  
                    "Ref": { "type": "string" },  
                    "Account": { "type": "string" },  
                    "LineStatus": { "type": "string" }  
                }  
            }  
        },  
        "TotalAmt": { "type": "number" }  
    }  
}
```

Modelo de mapeamento de saída (Exemplo de fatura de vendas)

Veja a seguir o modelo de mapeamento de saída que corresponde ao formato de dados JSON transformado. Aqui, as variáveis de modelo se baseiam no formato de dados JSON original, não transformado:

```
#set($inputRoot = $input.path('$'))
{
    "DueDate": "$inputRoot.DueDate",
    "Balance": "$inputRoot.Balance",
    "DocNumber": "$inputRoot.DocNumber",
    "Status": "$inputRoot.Status",
    "Line": [
        #foreach($elem in $inputRoot.Line)
        {
            "Description": "$elem.Description",
            "Amount": $elem.Amount,
            "DetailType": "$elem.DetailType",
            "Customer": "$elem.ExpenseDetail.Customer.value ($elem.ExpenseDetail.Customer.name)",
            "Ref": "$elem.ExpenseDetail.Ref.value ($elem.ExpenseDetail.Ref.name)",
            "Account": "$elem.ExpenseDetail.Account.value ($elem.ExpenseDetail.Account.name)",
            "LineStatus": "$elem.ExpenseDetail.LineStatus"
        }#if($foreach.hasNext),#end
    ]
},
    "TotalAmt": $inputRoot.TotalAmt
}
```

Exemplo de registro de funcionário (modelos e modelos de mapeamento do API Gateway)

As seções a seguir fornecem exemplos de modelos e modelos de mapeamento que podem ser usados para uma amostra de API de registro de funcionário no API Gateway. Para obter mais informações sobre modelos e modelos de mapeamento no API Gateway, consulte [Criar modelos e modelos de mapeamento para mapeamentos de solicitação e resposta \(p. 273\)](#).

Tópicos

- [Dados originais \(exemplo de registro de funcionário\) \(p. 296\)](#)
- [Modelo de entrada \(exemplo de registro de funcionário\) \(p. 297\)](#)
- [Modelo de mapeamento de entrada \(Exemplo de registro de funcionário\) \(p. 298\)](#)
- [Dados transformados \(Exemplo de registro de funcionário\) \(p. 299\)](#)
- [Modelo de saída \(exemplo de registro de funcionário\) \(p. 299\)](#)
- [Modelo de mapeamento de saída \(Exemplo de registro de funcionário\) \(p. 300\)](#)

Dados originais (exemplo de registro de funcionário)

Os dados a seguir são os dados JSON originais para o exemplo de registro de funcionário:

```
{
    "QueryResponse": {
        "maxResults": "1",
        "startPosition": "1",
        "Employee": {
            "Organization": "false",
            "Title": "Mrs.",
            "GivenName": "Jane",
            "MiddleName": "Lane",
            "FamilyName": "Doe",
            "DisplayName": "Jane Lane Doe",
            "PrintOnCheckName": "Jane Lane Doe",
            "Active": "true",
            "PrimaryPhone": { "FreeFormNumber": "505.555.9999" },
        }
    }
}
```

```
"PrimaryEmailAddr": { "Address": "janedoe@example.com" },
"EmployeeType": "Regular",
"status": "Synchronized",
"Id": "ABC123",
"SyncToken": "1",
"MetaData": {
    "CreateTime": "2015-04-26T19:45:03Z",
    "LastUpdatedTime": "2015-04-27T21:48:23Z"
},
"PrimaryAddr": {
    "Line1": "123 Any Street",
    "City": "Any City",
    "CountrySubDivisionCode": "WA",
    "PostalCode": "01234"
}
},
"time": "2015-04-27T22:12:32.012Z"
}
```

Modelo de entrada (exemplo de registro de funcionário)

O modelo de entrada a seguir corresponde aos dados JSON originais para o exemplo de registro de funcionário:

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "EmployeeInputModel",
    "type": "object",
    "properties": {
        "QueryResponse": {
            "type": "object",
            "properties": {
                "maxResults": { "type": "string" },
                "startPosition": { "type": "string" },
                "Employee": {
                    "type": "object",
                    "properties": {
                        "Organization": { "type": "string" },
                        "Title": { "type": "string" },
                        "GivenName": { "type": "string" },
                        "MiddleName": { "type": "string" },
                        "FamilyName": { "type": "string" },
                        "DisplayName": { "type": "string" },
                        "PrintOnCheckName": { "type": "string" },
                        "Active": { "type": "string" },
                        "PrimaryPhone": {
                            "type": "object",
                            "properties": {
                                "FreeFormNumber": { "type": "string" }
                            }
                        },
                        "PrimaryEmailAddr": {
                            "type": "object",
                            "properties": {
                                "Address": { "type": "string" }
                            }
                        },
                        "EmployeeType": { "type": "string" },
                        "status": { "type": "string" },
                        "Id": { "type": "string" },
                        "SyncToken": { "type": "string" },
                        "MetaData": {
                            "type": "object",
                            "properties": {
                                "CreateTime": { "type": "string" },
                                "LastUpdatedTime": { "type": "string" }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

```
        "properties": {
            "CreateTime": { "type": "string" },
            "LastUpdatedTime": { "type": "string" }
        }
    },
    "PrimaryAddr": {
        "type": "object",
        "properties": {
            "Line1": { "type": "string" },
            "City": { "type": "string" },
            "CountrySubDivisionCode": { "type": "string" },
            "PostalCode": { "type": "string" }
        }
    }
}
},
"time": { "type": "string" }
}
```

Modelo de mapeamento de entrada (Exemplo de registro de funcionário)

O modelo de mapeamento de entrada a seguir corresponde aos dados JSON originais para o exemplo de registro de funcionário:

```
#set($inputRoot = $input.path('$'))
{
    "QueryResponse": {
        "maxResults": "$inputRoot.QueryResponse.maxResults",
        "startPosition": "$inputRoot.QueryResponse.startPosition",
        "Employee": {
            "Organization": "$inputRoot.QueryResponse.Employee.Organization",
            "Title": "$inputRoot.QueryResponse.Employee.Title",
            "GivenName": "$inputRoot.QueryResponse.Employee.GivenName",
            "MiddleName": "$inputRoot.QueryResponse.Employee.MiddleName",
            "FamilyName": "$inputRoot.QueryResponse.Employee.FamilyName",
            "DisplayName": "$inputRoot.QueryResponse.Employee.DisplayName",
            "PrintOnCheckName": "$inputRoot.QueryResponse.Employee.PrintOnCheckName",
            "Active": "$inputRoot.QueryResponse.Employee.Active",
            "PrimaryPhone": { "FreeFormNumber": "$inputRoot.QueryResponse.Employee.PrimaryPhone.FreeFormNumber" },
            "PrimaryEmailAddr": { "Address": "$inputRoot.QueryResponse.Employee.PrimaryEmailAddr.Address" },
            "EmployeeType": "$inputRoot.QueryResponse.Employee.EmployeeType",
            "status": "$inputRoot.QueryResponse.Employee.status",
            "Id": "$inputRoot.QueryResponse.Employee.Id",
            "SyncToken": "$inputRoot.QueryResponse.Employee.SyncToken",
            "MetaData": {
                "CreateTime": "$inputRoot.QueryResponse.Employee.MetaData.CreateTime",
                "LastUpdatedTime": "$inputRoot.QueryResponse.Employee.MetaData.LastUpdatedTime"
            },
            "PrimaryAddr" : {
                "Line1": "$inputRoot.QueryResponse.Employee.PrimaryAddr.Line1",
                "City": "$inputRoot.QueryResponse.Employee.PrimaryAddr.City",
                "CountrySubDivisionCode": "$inputRoot.QueryResponse.Employee.PrimaryAddr.CountrySubDivisionCode",
                "PostalCode": "$inputRoot.QueryResponse.Employee.PrimaryAddr.PostalCode"
            }
        },
        "time": "$inputRoot.time"
    }
}
```

Dados transformados (Exemplo de registro de funcionário)

O exemplo a seguir mostra como os dados JSON do exemplo de registro de funcionário original podem ser transformados para saída:

```
{  
    "QueryResponse": {  
        "maxResults": "1",  
        "startPosition": "1",  
        "Employees": [  
            {  
                "Title": "Mrs.",  
                "GivenName": "Jane",  
                "MiddleName": "Lane",  
                "FamilyName": "Doe",  
                "DisplayName": "Jane Lane Doe",  
                "PrintOnCheckName": "Jane Lane Doe",  
                "Active": "true",  
                "PrimaryPhone": "505.555.9999",  
                "Email": [  
                    {  
                        "type": "primary",  
                        "Address": "janedoe@example.com"  
                    }  
                ],  
                "EmployeeType": "Regular",  
                "PrimaryAddr": {  
                    "Line1": "123 Any Street",  
                    "City": "Any City",  
                    "CountrySubDivisionCode": "WA",  
                    "PostalCode": "01234"  
                }  
            }  
        ],  
        "time": "2015-04-27T22:12:32.012Z"  
    }  
}
```

Modelo de saída (exemplo de registro de funcionário)

Veja a seguir o modelo de saída que corresponde ao formato de dados JSON transformado:

```
{  
    "$schema": "http://json-schema.org/draft-04/schema#",  
    "title": "EmployeeOutputModel",  
    "type": "object",  
    "properties": {  
        "QueryResponse": {  
            "type": "object",  
            "properties": {  
                "maxResults": { "type": "string" },  
                "startPosition": { "type": "string" },  
                "Employees": {  
                    "type": "array",  
                    "items": {  
                        "type": "object",  
                        "properties": {  
                            "Title": { "type": "string" },  
                            "GivenName": { "type": "string" },  
                            "MiddleName": { "type": "string" },  
                            "FamilyName": { "type": "string" },  
                            "DisplayName": { "type": "string" },  
                            "PrintOnCheckName": { "type": "string" },  
                        }  
                    }  
                }  
            }  
        }  
    }  
}
```

```
        "Active": { "type": "string" },
        "PrimaryPhone": { "type": "string" },
        "Email": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "type": { "type": "string" },
                    "Address": { "type": "string" }
                }
            }
        },
        "EmployeeType": { "type": "string" },
        "PrimaryAddr": {
            "type": "object",
            "properties": {
                "Line1": { "type": "string" },
                "City": { "type": "string" },
                "CountrySubDivisionCode": { "type": "string" },
                "PostalCode": { "type": "string" }
            }
        }
    }
},
"time": { "type": "string" }
}
```

Modelo de mapeamento de saída (Exemplo de registro de funcionário)

Veja a seguir o modelo de mapeamento de saída que corresponde ao formato de dados JSON transformado. Aqui, as variáveis de modelo se baseiam no formato de dados JSON original, não transformado:

```
#set($inputRoot = $input.path('$'))
{
    "QueryResponse": {
        "maxResults": "$inputRoot.QueryResponse.maxResults",
        "startPosition": "$inputRoot.QueryResponse.startPosition",
        "Employees": [
            {
                "Title": "$inputRoot.QueryResponse.Employee.Title",
                "GivenName": "$inputRoot.QueryResponse.Employee.GivenName",
                "MiddleName": "$inputRoot.QueryResponse.Employee.MiddleName",
                "FamilyName": "$inputRoot.QueryResponse.Employee.FamilyName",
                "DisplayName": "$inputRoot.QueryResponse.Employee.DisplayName",
                "PrintOnCheckName": "$inputRoot.QueryResponse.Employee.PrintOnCheckName",
                "Active": "$inputRoot.QueryResponse.Employee.Active",
                "PrimaryPhone": "$inputRoot.QueryResponse.Employee.PrimaryPhone.FreeFormNumber",
                "Email" : [
                    {
                        "type": "primary",
                        "Address": "$inputRoot.QueryResponse.Employee.PrimaryEmailAddr.Address"
                    }
                ],
                "EmployeeType": "$inputRoot.QueryResponse.Employee.EmployeeType",
                "PrimaryAddr": {
                    "Line1": "$inputRoot.QueryResponse.Employee.PrimaryAddr.Line1",
                    "City": "$inputRoot.QueryResponse.Employee.PrimaryAddr.City",
                    "CountrySubDivisionCode": "$inputRoot.QueryResponse.Employee.PrimaryAddr.CountrySubDivisionCode",
                }
            }
        ]
    }
}
```

```
        "PostalCode": "$inputRoot.QueryResponse.Employee.PrimaryAddr.PostalCode"
    }
}
],
"time": "$inputRoot.time"
}
```

Referência de mapeamento de dados de solicitações e respostas de API do Amazon API Gateway

Esta seção explica como configurar mapeamentos de dados a partir dos dados da solicitação de método de uma API, incluindo outros dados armazenados em variáveis [context \(p. 306\)](#), [stage \(p. 315\)](#) ou [util \(p. 315\)](#), para os parâmetros de solicitação de integração correspondentes e a partir dos dados de uma resposta de integração, incluindo os outros dados, para os parâmetros de resposta de método. Os dados de solicitação do método incluem parâmetros de solicitação (caminho, string de consulta, cabeçalhos) e o corpo. Os dados de resposta de integração incluem parâmetros de resposta (cabeçalhos) e o corpo. Para obter mais informações sobre o uso de variáveis de estágio, consulte [Referência a variáveis de estágio do Amazon API Gateway \(p. 546\)](#).

Tópicos

- [Mapear dados de solicitação de método para parâmetros de solicitação de integração \(p. 301\)](#)
- [Mapear dados de resposta de integração para cabeçalhos de resposta de método \(p. 303\)](#)
- [Mapear cargas de solicitação e resposta cargas entre método e integração \(p. 303\)](#)
- [Comportamentos de passagem direta de integração \(p. 304\)](#)

Mapear dados de solicitação de método para parâmetros de solicitação de integração

Parâmetros de solicitação de integração, no formato de variáveis de caminho, strings de consulta ou cabeçalhos, podem ser mapeados a partir de qualquer parâmetro de solicitação de método definido e da carga.

Na tabela a seguir, **PARAM_NAME** é o nome de um parâmetro de solicitação de método de tipo de parâmetro especificado. Deve corresponder à expressão regular `^[a-zA-Z0-9._$-]+$`. Ele deve ter sido definido antes de poder ser referenciado. **JSONPath_EXPRESSION** é uma expressão JSONPath para um campo JSON do corpo de uma solicitação ou resposta.

Note

O prefixo `$` é omitido nesta sintaxe.

Expressões de mapeamento de dados de solicitações de integração

Fonte de dados mapeada	Expressão de mapeamento
Caminho de solicitação de método	<code>method.request.path.PARAM_NAME</code>
String de consulta da solicitação de método	<code>method.request.querystring.PARAM_NAME</code>
String de consulta de solicitação do método de vários valores	<code>method.request.multivaluequerystring.PARAM_NAME</code>
Cabeçalho da solicitação de método	<code>method.request.header.PARAM_NAME</code>
Cabeçalho de solicitação de método de vários valores	<code>method.request.multivalueheader.PARAM_NAME</code>

Fonte de dados mapeada	Expressão de mapeamento
Corpo de solicitação de método	method.request.body
Corpo de solicitação de método (JsonPath)	method.request.body. JSONPath_EXPRESSION
Variáveis de estágio	stageVariables. VARIABLE_NAME
Variáveis de contexto	context. VARIABLE_NAME que deve ser uma das variáveis de contexto com suporte (p. 306).
Valor estático	' STATIC_VALUE '. STATIC_VALUE é um literal de string e deve estar entre aspas simples.

Example mapeamentos de parâmetros da mapeamento no OpenAPI

O exemplo a seguir mostra um trecho do OpenAPI que mapeia:

- o cabeçalho da solicitação de método, denominado `methodRequestHeaderParam`, no parâmetro do caminho de solicitação de integração, denominado `integrationPathParam`
- a string de consulta da solicitação de método de vários valores, denominada `methodRequestQueryParam`, na string de consulta da solicitação de integração, denominada `integrationQueryParam`

```

...
"requestParameters" : {
    "integration.request.path.integrationPathParam" :
    "method.request.header.methodRequestHeaderParam",
    "integration.request.querystring.integrationQueryParam" :
    "method.request.multivaluequerystring.methodRequestQueryParam"
}
...

```

Parâmetros de solicitação de integração também podem ser mapeados a partir de campos no corpo da solicitação JSON usando uma expressão `JSONPath`. A tabela a seguir mostra as expressões de mapeamento para um corpo de solicitação de método e seus campos JSON.

Example mapeamento do corpo da solicitação de método no OpenAPI

O exemplo a seguir mostra um trecho do OpenAPI que mapeia 1) o corpo da solicitação de método para o cabeçalho da solicitação de integração, denominado `body-header`, e 2) um campo JSON do corpo, conforme expresso por uma expressão JSON (`petstore.pets[0].name`, sem o prefixo `$.`).

```

...
"requestParameters" : {
    "integration.request.header.body-header" : "method.request.body",
    "integration.request.path.pet-name" : "method.request.body.petstore.pets[0].name",
}
...
```

Mapear dados de resposta de integração para cabeçalhos de resposta de método

Parâmetros de cabeçalho de resposta de método podem ser mapeados a partir de qualquer cabeçalho de resposta de integração ou corpo de resposta de integração, variáveis \$context ou valores estáticos.

Expressões de mapeamento do cabeçalho de resposta do método

Fonte de dados mapeada	Expressão de mapeamento
Cabeçalho da resposta de integração	<code>integration.response.header.PARAM_NAME</code>
Cabeçalho da resposta de integração	<code>integration.response.multivalueheader.PARAM_NAME</code>
Corpo da resposta de integração	<code>integration.response.body</code>
Corpo da resposta de integração (JsonPath)	<code>integration.response.body.JSONPath_EXPRESSION</code>
Variável de estágio	<code>stageVariables.VARIABLE_NAME</code>
Variável de contexto	<code>context.VARIABLE_NAME</code> que deve ser uma das variáveis de contexto com suporte (p. 306).
Valor estático	'STATIC_VALUE'. STATIC_VALUE é um literal de string e deve estar entre aspas simples.

Example mapeamento de dados a partir da resposta de integração no OpenAPI

O exemplo a seguir mostra um trecho do OpenAPI que mapeia 1) o campo JSONPath `redirect.url` da resposta de integração para o cabeçalho `location` da resposta de solicitação e 2) o cabeçalho `x-app-id` da resposta de integração para o cabeçalho `id` da resposta de método.

```
...
"responseParameters" : {
    "method.response.header.location" : "integration.response.body.redirect.url",
    "method.response.header.id" : "integration.response.header.x-app-id",
    "method.response.header.items" : "integration.response.multivalueheader.item",
}
...
```

Mapear cargas de solicitação e resposta cargas entre método e integração

O API Gateway usa o mecanismo [Velocity Template Language \(VTL\)](#) para processar [modelos de mapeamento](#) (p. 277) de corpo para a solicitação de integração e a resposta de integração. Os modelos de mapeamento convertem cargas de solicitação de método nas cargas de solicitação de integração correspondentes e convertem corpos de resposta de integração em corpos de resposta de método.

Os modelos VTL usam expressões JSONPath, outros parâmetros, como contextos de chamada e variáveis de estágio e funções de utilitário para processar os dados JSON.

Se um modelo estiver definido para descrever a estrutura de dados de uma carga, o API Gateway poderá usá-lo para gerar um esqueleto de modelo de mapeamento para uma solicitação de integração ou resposta de integração. Você pode usar esse esqueleto de modelo como ajuda para personalizar e

expandir o script de mapeamento VTL. No entanto, é possível criar um modelo de mapeamento do zero, sem definir um modelo para a estrutura de dados da carga.

Selecionar um modelo de mapeamento VTL

O API Gateway usa a seguinte lógica para selecionar um modelo de mapeamento, na especificação [VTL \(Velocity Template Language\)](#), para mapear a carga de uma solicitação de método para a solicitação de integração correspondente ou para mapear a carga de uma resposta de integração para a resposta de método correspondente.

Para uma carga de solicitação, o API Gateway usa o valor do cabeçalho `Content-Type` da solicitação como chave para selecionar o modelo de mapeamento para a carga de solicitação. Para uma carga de resposta, o API Gateway usa o valor do cabeçalho `Accept` da solicitação de entrada como chave para selecionar o modelo de mapeamento.

Quando o cabeçalho `Content-Type` está ausente na solicitação, o API Gateway pressupõe que o valor padrão seja `application/json`. Para tal solicitação, o API Gateway usa `application/json` como chave padrão para selecionar o modelo de mapeamento, se um estiver definido. Quando nenhum modelo corresponde a essa chave, o API Gateway transmitirá a carga não mapeada se a propriedade `passthroughBehavior` estiver definida como `WHEN_NO_MATCH` ou `WHEN_NO_TEMPLATES`.

Quando o cabeçalho `Accept` não está especificado na solicitação, o API Gateway pressupõe que o valor padrão seja `application/json`. Nesse caso, o API Gateway seleciona um modelo de mapeamento existente para `application/json` com o objetivo de mapear a carga de resposta. Se nenhum modelo estiver definido para `application/json`, o API Gateway selecionará o primeiro modelo existente e o usará como padrão para mapear a carga de resposta. Da mesma forma, o API Gateway usa o primeiro modelo existente quando o valor do cabeçalho `Accept` especificado não corresponde a nenhuma chave de modelo existente. Se nenhum modelo estiver definido, o API Gateway simplesmente transmitirá a carga da resposta não mapeada.

Por exemplo, suponha que uma API tenha um modelo `application/json` definido para uma carga de solicitação e tenha um modelo `application/xml` definido para a carga de resposta. Se o cliente definir os cabeçalhos "Content-Type : application/json" e "Accept : application/xml" na solicitação, ambas as cargas de solicitação e resposta serão processadas com os modelos de mapeamento correspondentes. Se o cabeçalho `Accept:application/xml` estiver ausente, o modelo de mapeamento `application/xml` será usado para mapear a carga de resposta. Em vez disso, para retornar a carga de resposta não mapeada, você deve configurar um modelo vazio para `application/json`.

Apenas o tipo MIME é usado nos cabeçalhos `Accept` e `Content-Type` ao selecionar um modelo de mapeamento. Por exemplo, um cabeçalho de "Content-Type: application/json; charset=UTF-8" terá um modelo de solicitação com a chave `application/json` selecionada.

Comportamentos de passagem direta de integração

Com integrações não proxy, quando uma solicitação de método contém uma carga, e o cabeçalho `Content-Type` não corresponde a nenhum modelo de mapeamento especificado ou nenhum modelo de mapeamento está definido, você pode optar por repassar a carga da solicitação fornecida pelo cliente por meio da solicitação de integração para o back-end sem transformação. O processo é conhecido como passagem direta de integração.

Para [integrações de proxy \(p. 222\)](#), o API Gateway repassa toda a solicitação para o back-end, e você não tem a opção de modificar os comportamentos de passagem.

O comportamento de passagem direta real de uma solicitação de entrada é determinado pela opção que você escolhe para um modelo de mapeamento especificado, durante a [configuração da solicitação de integração \(p. 270\)](#), e pelo cabeçalho `Content-Type` que um cliente define na solicitação de entrada. Os exemplos a seguir ilustram os possíveis comportamentos de passagem direta.

Exemplo 1: um modelo de mapeamento é definido na solicitação de integração para o tipo de conteúdo `application/json`.

Cabeçalho Content-Type\Opção de passagem direta selecionada	WHEN_NO_MATCH	WHEN_NO_TEMPLATE	NEVER
Nenhum (padrão para application/json)	A carga da solicitação é transformada usando o modelo.	A carga da solicitação é transformada usando o modelo.	A carga da solicitação é transformada usando o modelo.
application/json	A carga da solicitação é transformada usando o modelo.	A carga da solicitação é transformada usando o modelo.	A carga da solicitação é transformada usando o modelo.
application/xml	A carga da solicitação não é transformada e é enviada ao back-end no estado em que se encontra.	A solicitação é rejeitada com uma resposta HTTP 415 Unsupported Media Type.	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.

Cabeçalho Content-Type\Opção de passagem direta selecionada	WHEN_NO_MATCH	WHEN_NO_TEMPLATE	NEVER
Nenhum (padrão para application/json)	A carga da solicitação não é transformada e é enviada ao back-end no estado em que se encontra.	A solicitação é rejeitada com uma resposta HTTP 415 Unsupported Media Type.	A solicitação é rejeitada com uma resposta HTTP 415 Unsupported Media Type.
application/json	A carga da solicitação não é transformada e é enviada ao back-end no estado em que se encontra.	A solicitação é rejeitada com uma resposta HTTP 415 Unsupported Media Type.	A solicitação é rejeitada com uma resposta HTTP 415 Unsupported Media Type.
application/xml	A carga da solicitação é transformada usando o modelo.	A carga da solicitação é transformada usando o modelo.	A carga da solicitação é transformada usando o modelo.

Referência de variáveis de registro de acesso em logs e modelo de mapeamento do API Gateway

Esta seção fornece informações de referência para as variáveis e funções que o Amazon API Gateway define para uso com modelos de dados, autorizadores, modelos de mapeamento e registro de acesso em logs do CloudWatch. Para obter informações detalhadas sobre como usar essas variáveis e funções, consulte [the section called “Criar modelos e modelos de mapeamento” \(p. 273\)](#).

Tópicos

- [Variáveis \\$context para modelos de dados, autorizadores, modelos de mapeamento e registro de acesso em logs do CloudWatch \(p. 306\)](#)

- [Exemplo de modelo de variáveis \\$context \(p. 311\)](#)
- [Variáveis \\$context somente para registro de acesso em logs do CloudWatch \(p. 311\)](#)
- [Variáveis \\$input \(p. 312\)](#)
- [Exemplos de modelos de variáveis \\$input \(p. 313\)](#)
- [\\$stageVariables \(p. 315\)](#)
- [Variáveis \\$util \(p. 315\)](#)

Note

Para obter as variáveis \$method e \$integration, consulte [the section called “Referência de mapeamento de dados de solicitações e respostas” \(p. 301\)](#).

Variáveis \$context para modelos de dados, autorizadores, modelos de mapeamento e registro de acesso em logs do CloudWatch

As seguintes variáveis \$context podem ser usadas em modelos de dados, autorizadores, modelos de mapeamento e registro de acesso em logs do CloudWatch.

Para obter as variáveis \$context que podem ser usadas somente no registro de acesso em logs do CloudWatch, consulte [the section called “Variáveis \\$context somente para registro de acesso em logs do CloudWatch” \(p. 311\)](#).

Parâmetro	Descrição
\$context.accountId	O ID da conta da AWS do proprietário da API
\$context.apiId	O identificador que o API Gateway atribui à sua API.
\$context.authorizer.claims. <i>property</i>	Uma propriedade das declarações retornadas do grupo de usuários do Amazon Cognito depois que o agente de chamada do método é autenticado com êxito. Para obter mais informações, consulte the section called “Use o Grupo de usuários do Cognito como um autorizador para uma API REST” (p. 414) . Note Chamar \$context.authorizer.claims retorna um valor nulo.
\$context.authorizer.principalId	A identificação do usuário principal associada ao token enviado pelo cliente e retornado do autorizador do Lambda do API Gateway (anteriormente conhecido como autorizador personalizado). Para obter mais informações, consulte the section called “Usar autorizadores do Lambda” (p. 396) .
\$context.authorizer. <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:

Parâmetro	Descrição
	<pre>"context" : { "key": "value", "numKey": 1, "boolKey": true }</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>Para obter mais informações, consulte the section called “Usar autorizadores do Lambda” (p. 396).</p>
<code>\$context.awsEndpointRequestId</code>	O ID da solicitação do endpoint da AWS.
<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> . Ele é muitas vezes usado como um chamador/identificador do cliente.
<code>\$context.error.message</code>	Uma string de 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 “Monitorar a execução da API WebSocket com CloudWatch” (p. 650) e the section called “Configurar respostas do gateway” (p. 263) .
<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 “Monitorar a execução da API WebSocket com CloudWatch” (p. 650) e the section called “Configurar respostas do gateway” (p. 263) .
<code>\$context.error.validationErrorString</code>	Uma string que contém uma mensagem de erro de validação detalhada.

Parâmetro	Descrição
<code>\$context.extendedRequestId</code>	O ID estendido que o API Gateway atribui à solicitação de API, que contém mais informações úteis para depuração/solução de problemas.
<code>\$context.httpMethod</code>	O método HTTP utilizado. Os valores válidos incluem: <code>DELETE</code> , <code>GET</code> , <code>HEAD</code> , <code>OPTIONS</code> , <code>PATCH</code> , <code>POST</code> e <code>PUT</code> .
<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 “Utilização de planos de uso com chaves de API” (p. 450) .
<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 do agente de chamada que está fazendo a solicitação.
<code>\$context.identity.cognitoAuthenticationProvider</code>	O provedor de autenticação do Amazon Cognito usado pelo agente de chamada que está fazendo a solicitação. Disponível apenas se a solicitação tiver sido assinada com as credenciais do Amazon Cognito. Para obter informações, consulte Uso de identidades federadas no Guia do desenvolvedor do Amazon Cognito.
<code>\$context.identity.cognitoAuthenticationType</code>	O tipo de autenticação do Amazon Cognito usado pelo agente de chamada que está fazendo a solicitação. Disponível apenas se a solicitação tiver sido assinada com as credenciais do Amazon Cognito.
<code>\$context.identity.cognitoIdentityId</code>	O ID de identidade do Amazon Cognito usado pelo agente de chamada que está fazendo a solicitação. Disponível apenas se a solicitação tiver sido assinada com as credenciais do Amazon Cognito.
<code>\$context.identity.cognitoIdentityPoolId</code>	O ID do grupo de identidades do Amazon Cognito usado pelo agente de chamada que está fazendo a solicitação. Disponível apenas se a solicitação tiver sido assinada com as credenciais do Amazon Cognito.
<code>\$context.identity.principalOrgId</code>	O ID da organização da AWS .

Parâmetro	Descrição
<code>\$context.identity.sourceIp</code>	O endereço IP de origem da conexão TCP que está fazendo a solicitação ao API Gateway. Warning Você não deve confiar nesse valor se houver qualquer chance de que o cabeçalho do <code>X-Forwarded-For</code> possa ser falsificado.
<code>\$context.identity.user</code>	O identificador principal do usuário que está fazendo a solicitação. Usado em autorizadores do Lambda. Para obter mais informações, consulte the section called “Saída de um autorizador do Lambda do Amazon API Gateway” (p. 408) .
<code>\$context.identity.userAgent</code>	O cabeçalho User-Agent do chamador da API.
<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.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, é <code>HTTP/1.1</code> .
<code>\$context.requestId</code>	O ID que o API Gateway atribui à solicitação de API.
<code>\$context.requestOverride.header.header_name</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 Como usar um modelo de mapeamento para substituir parâmetros de solicitação e resposta e códigos de status de API (p. 280) .
<code>\$context.requestOverride.path.path_name</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 Como usar um modelo de mapeamento para substituir parâmetros de solicitação e resposta e códigos de status de API (p. 280) .

Parâmetro	Descrição
<code>\$context.requestOverride.querystring.querystring</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 Como usar um modelo de mapeamento para substituir parâmetros de solicitação e resposta e códigos de status de API (p. 280) .
<code>\$context.responseOverride.header.header</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 Como usar um modelo de mapeamento para substituir parâmetros de solicitação e resposta e códigos de status de API (p. 280) .
<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 Como usar um modelo de mapeamento para substituir parâmetros de solicitação e resposta e códigos de status de API (p. 280) .
<code>\$context.requestTime</code>	O horário da solicitação CLF formatado (dd/MMM/yyyy:HH:mm:ss +-hhmm).
<code>\$context.requestTimeEpoch</code>	O horário da solicitação Epoch formatado.
<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 https://rest-api-id.execute-api.{region}.amazonaws.com/{stage}/root/child , o valor <code>\$context.resourcePath</code> é /root/child. Para obter mais informações, consulte TUTORIAL: Criar uma API com integração não proxy HTTP (p. 59) .
<code>\$context.stage</code>	O estágio de implantação da solicitação de API (por exemplo, Beta ou Prod).

Parâmetro	Descrição
<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 “Use o AWS WAF para proteger sua API contra explorações comuns da web” (p. 449) .
<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 “Use o AWS WAF para proteger sua API contra explorações comuns da web” (p. 449) .
<code>\$context.xrayTraceId</code>	O ID de rastreamento para o rastreamento do X-Ray. Para obter mais informações, consulte the section called “Configuração do AWS X-Ray” (p. 587) .

Exemplo de modelo de variáveis `$context`

Você pode usar variáveis `$context` em um modelo de mapeamento se o seu método de API repassa dados estruturados a um back-end que exige que os dados estejam em um formato específico.

O exemplo a seguir mostra um modelo de mapeamento que mapeia variáveis `$context` de entrada para variáveis de back-end com nomes um pouco diferentes em uma carga de solicitação de integração:

Note

Observe que uma das variáveis é uma chave de API. Este exemplo pressupõe que o método tem “exige a chave de API” habilitado.

```
{
    "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"
}
```

Variáveis `$context` somente para registro de acesso em logs do CloudWatch

As variáveis `$context` a seguir estão disponíveis somente para registro de acesso em logs do CloudWatch. Para obter mais informações, consulte [the section called “Configurar registro de API em logs do CloudWatch” \(p. 536\)](#). (Para APIs WebSocket, consulte [the section called “Monitorar a execução da API WebSocket com CloudWatch” \(p. 650\)](#).)

Parâmetro	Descrição
<code>\$context.authorizer.integrationLatency</code>	A latência de autorizador em ms.
<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, não pela função de back-end do Lambda.
<code>\$context.responseLatency</code>	A latência da resposta em ms.
<code>\$context.responseLength</code>	O tamanho da carga de resposta.
<code>\$context.status</code>	O status de resposta do método.

Variáveis `$input`

A variável `$input` representa os parâmetros e a carga de solicitação do método a serem processados por um modelo de mapeamento. Ela fornece quatro funções:

Variável e função	Descrição
<code>\$input.body</code>	Retorna a carga de solicitação bruta como uma string.
<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> retorna uma string JSON que representa a estrutura <code>pets</code>.</p> <p>Para obter mais informações sobre o JSONPath, consulte JSONPath ou JSONPath para Java.</p>
<code>\$input.params()</code>	Retorna um mapa de todos os parâmetros de solicitação.
<code>\$input.params(x)</code>	<p>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>.</p> <p>Usa uma string de expressão JSONPath (<code>x</code>) e retorna uma representação de objeto JSON do resultado. Isso permite que você acesse e manipule elementos da carga nativamente em Apache VTL (Velocity Template Language).</p> <p>Por exemplo, se a expressão <code>\$input.path('\$.pets')</code> retorna um objeto da seguinte forma:</p> <pre>[{ "id": 1, "type": "dog", }</pre>

Variável e função	Descrição
	<pre> "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 JSONPath ou JSONPath para Java.</p>

Exemplos de modelos de variáveis `$input`

Exemplo de modelo de mapeamento de parâmetros

O seguinte exemplo de mapeamento de parâmetros transmite todos os parâmetros, incluindo `path`, `querystring` e `header`, ao endpoint de integração por meio de uma carga JSON:

```

#set($allParams = $input.params())
{
    "params" : {
        #foreach($type in $allParams.keySet())
        #set($params = $allParams.get($type))
        "$type" : {
            #foreach($paramName in $params.keySet())
            "$paramName" : "$util.escapeJavaScript($params.get($paramName))"
            #if($foreach.hasNext),#end
            #end
        }
        #if($foreach.hasNext),#end
        #end
    }
}

```

Na verdade, esse modelo de mapeamento processa a saída de todos os parâmetros de solicitação na carga, conforme descrito da seguinte forma:

```

{
    "parameters" : {
        "path" : {
            "path_name" : "path_value",
            ...
        }
    }
    "header" : {
        "header_name" : "header_value",
        ...
    }
    "querystring" : {
        "querystring_name" : "querystring_value",
        ...
    }
}

```

```
    }
}
```

Você pode querer usar a variável `$input` para obter strings de consulta e o corpo da solicitação com ou sem o uso de modelos. Você também pode inserir o parâmetro e a carga ou uma subseção da carga na sua função Lambda. Os exemplos a seguir mostram como fazer isso.

Exemplo de modelo de mapeamento JSON usando `$input`

O exemplo a seguir mostra como usar um mapeamento para ler um nome da string de consulta e depois incluir todo o corpo POST em um elemento:

```
{
  "name" : "$input.params('name')",
  "body" : $input.json('')
}
```

Se a entrada JSON contiver caracteres sem escape que não podem ser analisados pelo JavaScript, uma resposta 400 poderá ser retornada. A aplicação de `$util.escapeJavaScript($input.json('$'))` acima assegurará que a entrada JSON possa ser analisada corretamente.

Exemplo de modelo de mapeamento usando `$input`

O exemplo a seguir mostra como transmitir uma expressão JSONPath ao método `json()`. Você também pode ler uma propriedade específica do seu objeto de corpo da solicitação usando um ponto (.), seguido pelo nome da sua propriedade:

```
{
  "name" : "$input.params('name')",
  "body" : $input.json('.mykey')
}
```

Se uma carga de solicitação de método contiver caracteres sem escape que não podem ser analisados pelo JavaScript, você poderá obter uma resposta 400. Nesse caso, é necessário chamar a função `$util.escapeJavaScript()` no modelo de mapeamento, conforme mostrado a seguir:

```
{
  "name" : "$input.params('name')",
  "body" : $util.escapeJavaScript($input.json('.mykey'))
}
```

Exemplo de solicitação e resposta usando `$input`

Veja a seguir um exemplo que usa todas as três funções:

Request Template (Modelo de solicitação):

```
Resource: /things/{id}

With input template:
{
  "id" : "$input.params('id')",
  "count" : "$input.path('.things').size()",
  "things" : $util.escapeJavaScript($input.json('.things'))
}

POST /things/abc
{
  "things" : {
```

```
        "1" : {},
        "2" : {},
        "3" : {}
    }
```

Resposta:

```
{
  "id": "abc",
  "count": "3",
  "things": {
    "1": {},
    "2": {},
    "3": {}
  }
}
```

Para ver mais exemplos de mapeamento, consulte [Criar modelos e modelos de mapeamento para mapeamentos de solicitação e resposta \(p. 273\)](#).

\$stageVariables

Variáveis de estágio podem ser usadas no mapeamento de parâmetro e modelos de mapeamento e como espaços reservados em ARNs e URLs usados em integrações de método. Para obter mais informações, consulte [the section called “Configurar variáveis de estágio para uma API REST” \(p. 540\)](#).

Sintaxe	Descrição
<code>\$stageVariables.<variable_name></code>	<code><variable_name></code> representa um nome de variável de estágio.
<code>\$stageVariables['<variable_name>']</code>	<code><variable_name></code> representa qualquer nome de variável de estágio.
<code>\$(stageVariables['<variable_name>'])</code>	<code><variable_name></code> representa qualquer nome de variável de estágio.

Variáveis \$util

A variável `$util` contém funções de utilitário para uso em modelos de mapeamento.

Note

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	Descrição
	<p>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(data).replaceAll("\\"', '')</pre>
<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'))) { "errorMessageObjKey2ArrVal" : \$errorMessageObj.key2.arr[0] }</pre> <p>Você receberá a seguinte saída:</p> <pre>{ "errorMessageObjKey2ArrVal" : 1 }</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.
<code>\$util.base64Decode()</code>	Decodifica os dados de uma string codificada em base64.

Oferecer suporte a cargas úteis binárias no API Gateway

No API Gateway, a solicitação e a resposta da API podem ter uma carga de texto ou binária. Uma carga de texto é uma string JSON codificada em UTF-8, e uma carga binária é diferente de uma carga de texto. A carga binária pode ser, por exemplo, um arquivo JPEG, um arquivo GZip ou um arquivo XML.

Por padrão, o API Gateway trata o corpo da mensagem como uma carga de texto e aplica qualquer modelo de mapeamento pré-configurado para transformar a string JSON. Se nenhum modelo de mapeamento for especificado, o API Gateway pode passar a carga de texto do endpoint de integração ou para ele sem modificação, desde que o comportamento de passagem esteja ativado no método da API. Para uma carga binária, o API Gateway simplesmente passa a mensagem no estado em que ela se encontra.

Para o API Gateway passar cargas binárias, adicione os tipos de mídia à lista [binaryMediaTypes](#) do recurso [RestApi](#) ou defina as propriedades [contentHandling](#) nos recursos [Integration](#) e [IntegrationResponse](#). O valor [contentHandling](#) pode ser `CONVERT_TO_BINARY`, `CONVERT_TO_TEXT` ou indefinido. 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.
- Defina a propriedade [contentHandling](#) do recurso [Integration](#) como `CONVERT_TO_BINARY` para que a carga da solicitação seja convertida de uma string codificada em Base64 em seu blob binário ou defina a propriedade como `CONVERT_TO_TEXT` para que a carga da solicitação seja convertida de um blob binário em uma string codificada em Base64. Se essa propriedade não estiver definida, o API Gateway transmitirá a carga sem modificação. Isso ocorre quando o valor do cabeçalho `Content-Type` corresponde a uma das entradas [binaryMediaTypes](#) e os [comportamentos de passagem direta](#) (p. 304) também são habilitados para a API.
- Defina a propriedade [contentHandling](#) do recurso [IntegrationResponse](#) como `CONVERT_TO_BINARY` para que a carga da resposta seja convertida de uma string codificada em Base64 em seu blob binário ou defina a propriedade como `CONVERT_TO_TEXT` para que a carga da resposta seja convertida de um blob binário em uma string codificada em Base64. Se [contentHandling](#) não for definido e se o cabeçalho `Content-Type` da resposta e o cabeçalho `Accept` da solicitação original corresponderem a uma entrada da lista [binaryMediaTypes](#), o API Gateway transmitirá o corpo. Isso ocorre quando o cabeçalho `Content-Type` e o cabeçalho `Accept` são os mesmos; caso contrário, o API Gateway converterá o corpo de resposta no tipo especificado no cabeçalho `Accept`.

Tip

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 de `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 `` 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.

Tópicos

- [Conversões de tipo de conteúdo no API Gateway](#) (p. 318)
- [Habilitar o suporte a binário usando o console do API Gateway](#) (p. 320)
- [Habilitar o suporte a binários usando o API REST do API Gateway](#) (p. 324)
- [Importar e exportar codificações de conteúdo](#) (p. 328)
- [Exemplos de suporte a binários](#) (p. 328)

Conversões de tipo de conteúdo no API Gateway

A tabela a seguir mostra como o API Gateway converte a carga 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.

Conversões de tipo de conteúdo de solicitação de API no API Gateway

Carga da solicitação de método	Cabeçalho Content-Type da solicitação	binaryMediaType	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
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 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.

Conversões de tipo de conteúdo de respostas do API Gateway

Carga da resposta de integração	Cabeçalho Accept da solicitação	binaryMediaType	contentHandling	Mapear a carga da resposta
Dados de texto ou binários	Um tipo de texto	Não definido	Não definido	String codificada em UTF8

Carga da resposta de integração	Cabeçalho Accept da solicitação	binaryMediaType	contentHandling	Mapear a carga da resposta
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
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

Tip

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 de `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 `` 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.

Ao converter uma carga de texto em um blob binário, o API Gateway pressupõe que os dados de texto sejam uma string de caracteres codificada 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 \(p. 304\)](#) na API.

Ao converter uma carga binária em uma string de texto, o API Gateway sempre aplica uma codificação em 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 nesse modelo de mapeamento via `$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 \(p. 304\)](#) na API.

Habilitar o suporte a binário usando o console do API Gateway

Esta seção explica como habilitar o suporte a binários usando o console do API Gateway. Como exemplo, usaremos uma API integrada ao Amazon S3. Nossa 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: Criar uma API REST como um proxy do Amazon S3 no API Gateway \(p. 116\)](#).

Para habilitar o suporte a 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 `FileManager`.
 - b. Na API selecionada no painel de navegação principal, escolha `Settings`.
 - c. No painel `Settings`, escolha `Add Binary Media Type` na seção `Binary Media Types`.
 - d. Digite 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 `/`.
 - e. Escolha `Save Changes` (Salvar alterações).

Settings

Configure settings for your API deployments

API Key Source

Choose the source of your API Keys from incoming requests. Configure deployments to receive API keys from the X-APIKEY_HEADER or from a Custom Authorizer

API Key Source

Content Encoding

Allow compression of response bodies based on client's Accept-Encoding header. Compression can be customized to be enabled above a certain threshold response body size. The following compression types are supported: gzip, deflate, and identity.

Content Encoding enabled

Binary Media Types

You can configure binary support for your API by specifying which media types should be treated as binary types. API Gateway will look at the Content-Type and Accept HTTP headers to decide how to handle the body.

2. Defina como as cargas de mensagem são manipuladas para o método de API:
 - a. Crie um novo recurso 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 Tratamento de conteúdo, escolha uma opção.

/{{folder}}/{{item}} - GET - Setup

Choose the integration point for your new method.

Integration type Lambda Function i

HTTP i

Mock i

AWS Service i

AWS Region

AWS Service

AWS Subdomain

HTTP method

Action Type Use action name

Use path override

Path override (optional)

Execution role

i

Content Handling

Save

Escolha Passagem se não desejar converter o corpo quando o cliente e o back-end aceitarem o mesmo formato binário. Escolha Converter em texto (se necessário) para converter o corpo do binário em uma string codificada em Base64 quando, por exemplo, o back-end exige que uma carga de solicitação binária seja transmitida como a propriedade JSON. E escolha Converter em binário (se necessário) quando o cliente enviar uma sequência codificada em Base64 e o back-end exigir o formato binário original, ou quando o endpoint retornar uma sequência codificada em Base64 e o cliente aceitar apenas a saída binária.

- d. 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

Name	Mapped from <small>i</small>	Caching	
Accept	method.request.header.Accept	<input type="checkbox"/>	 
Content-Type	method.request.header.Content-Type	<input type="checkbox"/>	 

 [Add header](#)

- e. Habilite o comportamento de passagem direta no corpo da solicitação.

▼ Body Mapping Templates

Request body passthrough When no template matches the request Content-Type header i

When there are no templates defined (recommended) i

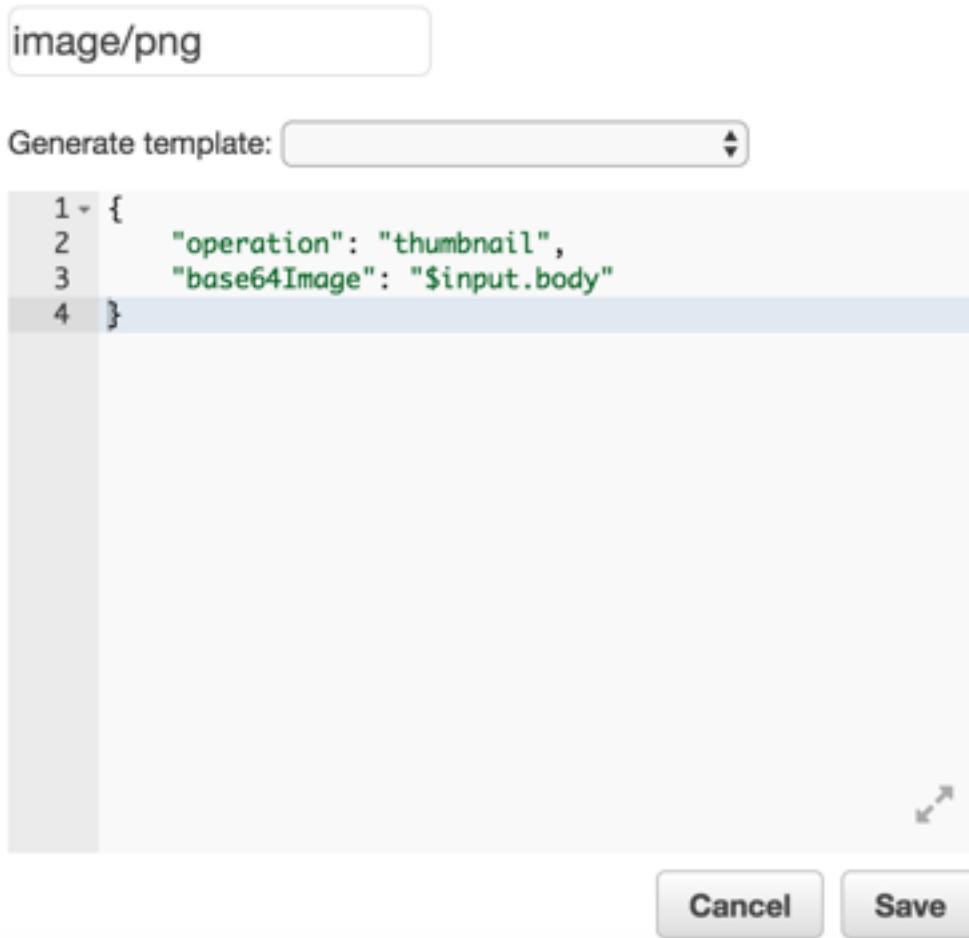
Never i

Content-Type

image/png

 [Add mapping template](#)

- f. Para conversão em texto, defina um modelo de mapeamento para colocar os dados binários codificados em Base64 no formato necessário.



O formato desse modelo de mapeamento depende das exigências de endpoint da entrada.

Habilitar o suporte a binários usando o API REST do API Gateway

As tarefas a seguir mostram como habilitar o suporte a binário usando chamadas de API REST do API Gateway.

Tópicos

- [Adicionar e atualizar tipos de mídia binários com suporte para uma API \(p. 325\)](#)
- [Configurar conversões de carga de solicitação \(p. 325\)](#)
- [Configurar conversões de carga de resposta \(p. 325\)](#)
- [Converter dados binários em dados de texto \(p. 326\)](#)
- [Converter dados de texto em uma carga binária \(p. 326\)](#)
- [Transmitir uma carga binária \(p. 327\)](#)

Adicionar e atualizar tipos de mídia binários 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 maneira.

```
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`, conforme mostrado a seguir:

```
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`, enviando uma solicitação `PATCH`, conforme mostrado a seguir:

```
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-octet-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 para 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-octet-stream",  
            "value" : "{\"body\": \"$input.body\"}"  
        }  
    ]  
}
```

Converter dados de texto em uma carga binária

Suponha que uma função 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-octet-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 invoca o modelo de passagem direta para retornar o blob binário decodificado em Base64 como arquivo de imagem para o 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-octet-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 direta.

```
PATCH /restapis/<restapi_id>/resources/<resource_id>/methods/<http_method>/integration

{
  "patchOperations" : [
    {
      "op" : "replace",
      "path" : "/contentHandling",
      "value" : "CONVERT_TO_BINARY"
    },
    {
      "op" : "replace",
      "path" : "/passthroughBehavior",
      "value" : "WHEN_NO_MATCH"
    }
  ]
}
```

```
        "path" : "/passthroughBehaviors",
        "value" : "WHEN_NO_MATCH"
    }
}
```

Importar e exportar codificações de conteúdo

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 \(p. 613\)](#)

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 \(p. 617\)](#)
- [Objeto x-amazon-apigateway-integration.response \(p. 624\)](#)

Exemplos de suporte a binários

O exemplo a seguir demonstra como acessar um arquivo binário no Amazon S3 ou no AWS Lambda por meio de uma API do API Gateway. A API de amostra é apresentada em um arquivo do OpenAPI. O exemplo de código usa as chamadas da API REST do API Gateway.

Tópicos

- [Acessar arquivos binários no Amazon S3 por meio de uma API do API Gateway \(p. 328\)](#)
- [Acessar arquivos binários no Lambda usando uma API do API Gateway \(p. 334\)](#)

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 amostra para acessar imagens no Amazon S3 \(p. 328\)](#)
- [Fazer download de uma imagem do Amazon S3 \(p. 332\)](#)
- [Carregar uma imagem para o Amazon S3 \(p. 333\)](#)

Arquivo do OpenAPI de uma API de amostra 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, 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"
          }
        ]
      }
    }
  }
}
```

```
        "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\w{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 sequência codificada em Base64, formatado como uma propriedade JSON, no Amazon S3, adicione um modelo de resposta para a resposta de integração 200 como esta, conforme mostrado no bloco de definição de OpenAPI em negrito a seguir:

```
"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 baixar o arquivo de imagem é semelhante a esta:

```
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 fazer upload de 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 fazer upload de 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=

Observe que 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 a seguir demonstra como acessar um arquivo binário no AWS Lambda por meio de uma API do API Gateway. A API de amostra é apresentada em um arquivo do OpenAPI. O exemplo de código usa as chamadas da API REST do API Gateway.

Tópicos

- [Arquivo do OpenAPI de uma API de amostra para acessar imagens no Lambda \(p. 334\)](#)
- [Fazer download de uma imagem do Lambda \(p. 338\)](#)
- [Carregar uma imagem para o Lambda \(p. 338\)](#)

Arquivo do OpenAPI de uma API de amostra 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\": \"$input.params('key')\"\n        }"
    },
    "responses": {
        "default": {
            "statusCode": "500"
        },
        "2\d{2)": {
            "statusCode": "200",
            "responseTemplates": {
                "application/json": "{\n                    \"image\": \"$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        }",
        "responses": {
            "default": {

```

```
        "statusCode": "500"
    },
    "2\\d{2}": {
        "statusCode": "200"
    }
}
}
},
"x-amazon-apigateway-binary-media-types": [
    "application/octet-stream",
    "image/jpeg"
],
"servers": [
    {
        "url": "https://abcdefghi.execute-api.us-east-1.amazonaws.com/{basePath}",
        "variables": {
            "basePath": {
                "default": "/v1"
            }
        }
    }
],
"components": {
    "schemas": {
        "Empty": {
            "type": "object",
            "title": "Empty Schema"
        }
    }
}
}
```

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "version": "2016-10-21T17:26:28Z",
    "title": "ApiName"
  },
  "host": "abcdefghi.execute-api.us-east-1.amazonaws.com",
  "basePath": "/v1",
  "schemes": [
    "https"
  ],
  "paths": {
    "/lambda": {
      "get": {
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "key",
            "in": "query",
            "required": false,
            "type": "string"
          }
        ],
        "responses": {
          "200": {
            "description": "200 response",
            "schema": {

```

```
        "$ref": "#/definitions/Empty"
    },
},
"500": {
    "description": "500 response"
}
},
"x-amazon-apigateway-integration": {
    "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789012:function:image/invocations",
    "type": "AWS",
    "credentials": "arn:aws:iam::123456789012:role/Lambda",
    "httpMethod": "POST",
    "requestTemplates": {
        "application/json": "{\n          \"imageKey\": \"$input.params('key')\"\n        }"
    },
    "responses": {
        "default": {
            "statusCode": "500"
        },
        "2\\\d{2}": {
            "statusCode": "200",
            "responseTemplates": {
                "application/json": "{\n          \"image\": \"$input.body\"\n        }"
            }
        }
    }
},
"put": {
    "produces": [
        "application/json", "application/octet-stream"
    ],
    "parameters": [
        {
            "name": "key",
            "in": "query",
            "required": false,
            "type": "string"
        }
    ],
    "responses": {
        "200": {
            "description": "200 response",
            "schema": {
                "$ref": "#/definitions/Empty"
            }
        },
        "500": {
            "description": "500 response"
        }
    },
    "x-amazon-apigateway-integration": {
        "uri": "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:123456789012:function:image/invocations",
        "type": "AWS",
        "credentials": "arn:aws:iam::123456789012:role/Lambda",
        "httpMethod": "POST",
        "contentHandling": "CONVERT_TO_TEXT",
        "requestTemplates": {
            "application/json": "{\n              \"imageKey\": \"$input.params('key')\", \"image\":
$\"input.body\"\n            }"
        },
        "responses": {
            "default": {
                "statusCode": "500"
            }
        }
    }
}
```

```
        },
        "2\\d{2}": {
            "statusCode": "200"
        }
    }
},
"x-amazon-apigateway-binary-media-types" : ["application/octet-stream", "image/jpeg"],
"definitions": {
    "Empty": {
        "type": "object",
        "title": "Empty Schema"
    }
}
}
```

Fazer download de uma imagem do Lambda

Para fazer download de 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 fazer download de 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 fazer upload de 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 fazer upload de 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

Habilitar compactação de carga para uma API

O API Gateway permite que o cliente chame uma API com cargas compactadas usando uma das [codificações de conteúdo compatíveis \(p. 341\)](#). 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 `minimumCompressionSize` como um inteiro não negativo entre 0 e 10485760 (10 milhões de bytes) quando criar a API ou depois de criar a API. Para desabilitar a compactação na API, defina `minimumCompressionSize` como nulo ou removê-lo por completo. Você pode habilitar ou desabilitar a compactação para 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 pode 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 a carga de solicitação do método e aplicar 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 compactação de carga para uma API \(p. 340\)](#)
- [Como chamar um método de API com carga compactada \(p. 341\)](#)
- [Receber resposta da API com uma carga compactada \(p. 342\)](#)

Habilitar compactação de carga para uma API

Você pode habilitar a compressão de uma API usando o console do API Gateway, a AWS CLI ou o AWS SDK.

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 compactação de carga para uma API usando o console do API Gateway \(p. 340\)](#)
- [Habilitar compactação de carga para uma API usando a AWS CLI \(p. 340\)](#)
- [Codificações de conteúdo com suporte do API Gateway \(p. 341\)](#)

Habilitar 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.

Para habilitar a compactação de carga usando o console do API Gateway

1. Faça login no console do API Gateway.
2. Escolha uma API existente ou crie uma nova.
3. No painel de navegação principal, escolha Settings sob a API que você escolheu ou criou.
4. Na seção Content Encoding do painel Settings, selecione a opção Content Encoding enabled para habilitar a compactação de carga. Digite um número para o tamanho mínimo de compactação (em bytes) próximo a Minimum body size required for compression. Para desabilitar a compactação, desmarque a opção Content Encoding enabled.
5. Selecione Save Changes (Salvar alterações).

Habilitar compactação de carga para uma API usando a AWS CLI

Para usar a AWS CLI a fim de criar uma nova API e habilitar a compactação, chame o comando `create-rest-api` da seguinte forma:

```
aws apigateway create-rest-api \
--name "My test API" \
--minimum-compression-size 0
```

Para usar a AWS CLI a fim de habilitar a compactação de uma API existente, chame o comando `update-rest-api` da seguinte forma:

```
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.

Para usar a AWS CLI a fim de desabilitar a compactação, chame o comando `update-rest-api` da seguinte forma:

```
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.

Codificações de conteúdo com suporte do API Gateway

O API Gateway oferece suporte às seguintes codificações de conteúdo:

- `deflate`
- `gzip`
- `identity`

O API Gateway também oferece suporte ao 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=1.0`
- `Accept-Encoding: gzip;q=1.0,identity;q=0.5,*;q=0`

Como chamar um método de API com carga compactada

Para fazer uma solicitação da API com uma carga compactada, o cliente deve definir o cabeçalho `Content-Encoding` com uma das [codificações de conteúdo com suporte \(p. 341\)](#).

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#OW##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 no

CloudWatch Logs se ele ocorrer na fase inicial de descompactação antes de a sua API e etapa serem identificadas.

Receber resposta da API com uma carga compactada

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 \(p. 341\)](#).

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 compactará 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 acionar a compactação de carga 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 contém um cabeçalho `Content-Encoding`, o API Gateway pressupõe que a carga da resposta de integração já está compactada e ignora o processamento de compactação.

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 back-end 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#)JV
#:P^IeA*#####+ (#L #X#YZ#kuOL0B7!9##C##&####Y##a###^#X
```

Quando a carga de resposta é compactada, a cobrança pela transferência de dados leva em conta apenas o tamanho dos dados compactados.

Ativar a validação de solicitação 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 chamador e publica os resultados da validação em logs do CloudWatch. Isso reduz as chamadas desnecessárias para o back-end. O mais importante, isso permite concentrar-se nos esforços de validação específicos para o seu aplicativo.

Tópicos

- Visão geral da validação básica de solicitações no API Gateway (p. 343)
- Configurar a validação básica de solicitações no API Gateway (p. 344)
- Testar a validação básica de solicitações no API Gateway (p. 349)
- Definições do OpenAPI de uma API de amostra com a validação básica de solicitações (p. 353)

Visão geral da validação básica de solicitações no API Gateway

O API Gateway pode executar a validação básica. Isso permite que você, o desenvolvedor de APIs, concentre-se na validação profunda específica do aplicativo no back-end. Para a validação básica, 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.
- Se a carga de solicitações aplicável atende ao [modelo \(p. 279\)](#) de solicitação do [esquema JSON](#) do método.

Para habilitar a validação básica, 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

A validação do corpo da solicitação e a [passagem do corpo da solicitação \(p. 304\)](#) são dois problemas separados. Quando uma carga de solicitação não pode ser validada porque nenhum esquema de modelo pode ser correspondido, você pode escolher passar ou bloquear a carga

original. Por exemplo, quando você habilita a validação da solicitação com um modelo de mapeamento para o tipo de mídia `application/json`, você pode passar uma carga XML por meio do back-end, embora a validação da solicitação habilitada falhe. Esse poderá ser o caso se você pretender oferecer suporte à carga XML no futuro. Para que uma solicitação falhe com uma carga XML, você deve escolher explicitamente a opção `NEVER` para o comportamento de passagem de conteúdo.

Configurar a validação básica de solicitações no API Gateway

Você pode configurar validadores de solicitação no arquivo de definição do OpenAPI de uma API e depois importar as definições do OpenAPI para o API Gateway. Você também pode configurá-los no console do API Gateway ou chamando a API REST do API Gateway, a CLI da AWS ou um dos SDKs da AWS. Aqui, mostramos como fazer isso com um arquivo do OpenAPI, no console e usando a API REST do API Gateway.

Tópicos

- [Configurar a validação básica de solicitações importando a definição do OpenAPI \(p. 344\)](#)
- [Configurar validadores de solicitação usando a API REST do API Gateway \(p. 348\)](#)
- [Configurar a validação básica de solicitações usando o console do API Gateway \(p. 349\)](#)

Configurar a validação básica de solicitações importando a definição do OpenAPI

As etapas a seguir descrevem como habilitar a validação básica de solicitações importando um arquivo do OpenAPI.

Para habilitar a validação da solicitação importando um arquivo do OpenAPI no API Gateway

1. Declare validadores de solicitação no OpenAPI, especificando um conjunto de objetos [Objeto x-amazon-apigateway-request-validation \(p. 628\)](#) no mapa [Objeto x-amazon-apigateway-request-validation \(p. 627\)](#) em nível de API. Por exemplo, o [arquivo do OpenAPI de amostra da API \(p. 353\)](#) contém o mapa `x-amazon-apigateway-request-validation`, com os nomes dos validadores como chaves.

OpenAPI 3.0

```
{  
    "openapi": "3.0.0",  
    "info": {  
        "title": "ReqValidation Sample",  
        "version": "1.0.0"  
    },  
    "servers": [  
        {  
            "url": "/{basePath}",  
            "variables": {  
                "basePath": {  
                    "default": "/v1"  
                }  
            }  
        }  
    ],  
    "x-amazon-apigateway-request-validation": {  
        "all": {  
            "validateRequestBody": true,  
            "validateRequestParameters": true  
        },  
        "params-only": {  
            "validateRequestBody": false,  
            "validateRequestParameters": true  
        }  
    }  
}
```

```
        }
    }
}
```

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "title": "ReqValidation Sample",
    "version": "1.0.0"
  },
  "schemes": [
    "https"
  ],
  "basePath": "/v1",
  "produces": [
    "application/json"
  ],
  "x-amazon-apigateway-requestValidators": {
    "all": {
      "validateRequestBody": true,
      "validateRequestParameters": true
    },
    "params-only": {
      "validateRequestBody": false,
      "validateRequestParameters": true
    }
  },
  ...
}
```

Você seleciona o nome de um validador ao habilitar esse validador na API ou em um método, conforme mostrado na próxima etapa.

2. Para habilitar um validador de solicitação em todos os métodos de uma API, especifique uma propriedade [Propriedade x-amazon-apigateway-request-validator \(p. 626\)](#) no nível de API do arquivo de definição do OpenAPI. Para habilitar um validador de solicitação em um método individual, especifique a propriedade `x-amazon-apigateway-request-validator` no nível do método. Por exemplo, a seguinte propriedade `x-amazon-apigateway-request-validator` habilita o validador `params-only` em todos os métodos da API, a menos que ele seja substituído de outra forma.

OpenAPI 3.0

```
{
  "openapi": "3.0.0",
  "info": {
    "title": "ReqValidation Sample",
    "version": "1.0.0"
  },
  "servers": [
    {
      "url": "/{basePath}",
      "variables": {
        "basePath": {
          "default": "/v1"
        }
      }
    }
  ],
  "x-amazon-apigateway-requestValidator": "params-only",
  ...
}
```

```
}
```

OpenAPI 2.0

```
{  
    "swagger": "2.0",  
    "info": {  
        "title": "ReqValidation Sample",  
        "version": "1.0.0"  
    },  
    "schemes": [  
        "https"  
    ],  
    "basePath": "/v1",  
    "produces": [  
        "application/json"  
    ],  
    ...  
    "x-amazon-apigateway-request-validator" : "params-only",  
    ...  
}
```

Para habilitar um validador de solicitação em um método individual, especifique a propriedade `x-amazon-apigateway-request-validator` no nível do método. Por exemplo, a seguinte propriedade `x-amazon-apigateway-request-validator` habilita o validador `all` no método `POST /validation`. Isso substitui o validador `params-only` herdado da API.

OpenAPI 3.0

```
{  
    "openapi": "3.0.0",  
    "info": {  
        "title": "ReqValidation Sample",  
        "version": "1.0.0"  
    },  
    "servers": [  
        {  
            "url": "/{basePath}",  
            "variables": {  
                "basePath": {  
                    "default": "/v1"  
                }  
            }  
        }  
    ],  
    "paths": {  
        "/validation": {  
            "post": {  
                "x-amazon-apigateway-request-validator": "all",  
                ...  
            }  
        }  
    }  
}
```

OpenAPI 2.0

```
{  
    "swagger": "2.0",  
    ...  
}
```

```
"info": {  
    "title": "ReqValidation Sample",  
    "version": "1.0.0"  
},  
"schemes": [  
    "https"  
,  
    "basePath": "/v1",  
    "produces": [  
        "application/json"  
,  
        ...  
    ],  
    "paths": {  
        "/validation": {  
            "post": {  
                "x-amazon-apigateway-request-validator" : "all",  
                ...  
            },  
            ...  
        }  
    }  
}
```

3. No API Gateway, crie a API com validadores de solicitação habilitados, importando esta [definição do OpenAPI de exemplo \(p. 353\)](#):

```
POST /restapis?mode=import&failonwarning=true HTTP/1.1  
Content-Type: application/json  
Host: apigateway.us-east-1.amazonaws.com  
X-Amz-Date: 20170306T234936Z  
Authorization: AWS4-HMAC-SHA256 Credential={access_key_ID}/20170306/us-east-1/  
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,  
Signature={sig4_hash}
```

Copy the JSON object from this sample OpenAPI definition (p. 353) and paste it here.

4. Implante a API recém-criada (fjd6crafxc) em um estágio especificado (testStage).

```
POST /restapis/fjd6crafxc/deployments HTTP/1.1  
Content-Type: application/json  
Host: apigateway.us-east-1.amazonaws.com  
X-Amz-Date: 20170306T234936Z  
Authorization: AWS4-HMAC-SHA256 Credential={access_key_ID}/20170306/us-east-1/  
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,  
Signature={sig4_hash}  
  
{  
    "stageName" : "testStage",  
    "stageDescription" : "Test stage",  
    "description" : "First deployment",  
    "cacheClusterEnabled" : "false"  
}
```

Para obter instruções sobre como testar a validação da solicitação usando a API REST do API Gateway, consulte [Testar a validação básica de solicitações usando a API REST do API Gateway \(p. 350\)](#). Para obter instruções sobre como testar usando o console do API Gateway, consulte [Testar a validação básica de solicitações usando o console do API Gateway \(p. 352\)](#).

Configurar validadores de solicitação usando a API REST do API Gateway

Na API REST do API Gateway, um validador de solicitação é representado por um recurso [RequestValidator](#). Para que uma API ofereça suporte aos mesmos validadores de solicitação que a [API de Amostra \(p. 353\)](#), adicione à coleção [RequestValidators](#) um validador exclusivo de parâmetro com `params-only` como chave e adicione um validador completo com `all` como chave.

Para habilitar a validação básica de solicitações usando a API REST do API Gateway

Partimos do princípio de que você tenha uma API semelhante à [API de amostra \(p. 353\)](#), mas não tenha configurado os validadores de solicitação. Se a sua API já tem validadores de solicitações habilitados, chame a ação `requestvalidator:update` ou `method:put` apropriada em vez de `requestvalidator:create` ou `method:put`.

1. Para configurar o validador de solicitações `params-only`, chame a ação `requestvalidator:create`, da seguinte maneira:

```
POST /restapis/restapi-id/requestvalidators HTTP/1.1
Content-Type: application/json
Host: apigateway.region.amazonaws.com
X-Amz-Date: 20170223T172652Z
Authorization: AWS4-HMAC-SHA256 Credential={access_key_ID}/20170223/region/apigateway/
aws4_request, SignedHeaders=content-type;host;x-amz-date, Signature={sig4_hash}

{
  "name" : "params-only",
  "validateRequestBody" : "false",
  "validateRequestParameters" : "true"
}
```

2. Para configurar o validador de solicitações `all`, chame a ação `requestvalidator:create`, da seguinte maneira:

```
POST /restapis/restapi-id/requestvalidators HTTP/1.1
Content-Type: application/json
Host: apigateway.region.amazonaws.com
X-Amz-Date: 20170223T172652Z
Authorization: AWS4-HMAC-SHA256 Credential={access_key_ID}/20170223/region/apigateway/
aws4_request, SignedHeaders=content-type;host;x-amz-date, Signature={sig4_hash}

{
  "name" : "all",
  "validateRequestBody" : "true",
  "validateRequestParameters" : "true"
}
```

Se as chaves de validador anteriores já existirem no mapa `RequestValidators`, chame a ação `requestvalidator:update` em vez para redefinir as regras de validação.

3. Para aplicar o validador de solicitação `all` ao método `POST`, chame `method:put` para habilitar o validador especificado (conforme identificado pela propriedade `requestValidatorId`) ou chame `method:update` para atualizar o validador habilitado.

```
PUT /restapis/restapi-id/resources/resource-id/methods/POST HTTP/1.1
Content-Type: application/json
Host: apigateway.region.amazonaws.com
X-Amz-Date: 20170223T172652Z
Authorization: AWS4-HMAC-SHA256 Credential={access_key_ID}/20170223/region/apigateway/
aws4_request, SignedHeaders=content-type;host;x-amz-date, Signature={sig4_hash}

{
```

```
    "authorizationType" : "NONE",
    ...
    "requestValidatorId" : "all"
}
```

Configurar a validação básica de solicitações usando o console do API Gateway

O console do API Gateway permite que você configure a validação básica de solicitações em um método usando um dos três validadores:

- Validate body: Este é o validador apenas do corpo.
- Validate query string parameters and headers: Este é o validador apenas para parâmetros.
- Validate body, query string parameters, and headers: Este validador é para ambos os parâmetros de corpo e validação.

Quando você escolher um dos validadores acima para habilitá-lo em um método de API, o console do API Gateway adicionará o validador ao mapa [RequestValidators](#) da API, se esse validador ainda não tiver sido adicionado ao mapa de validadores da API.

Para habilitar um validador de solicitações em um método

1. Entre no console do API Gateway caso ainda não esteja conectado.
2. Crie uma nova ou escolha uma API existente.
3. Crie um novo recurso ou escolha um recurso existente da API.
4. Crie um novo método ou escolha um método existente do recurso.
5. Escolha Method Request (Solicitação de método).
6. Escolha o ícone de lápis Request Validator em Settings.
7. Escolha Validate body, Validate query string parameters and headers ou Validate body, query string parameters, and headers em Request Validator na lista suspensa e depois escolha o ícone de marca de seleção para salvar sua opção.

The screenshot shows the 'Method Execution' settings for a 'GET - Method Request'. It includes fields for Authorization (set to 'NONE'), Request Validator (set to 'Validate query strings parameters and headers'), and API Key Required (set to 'false'). There are also tabs for 'Settings' and 'Test'.

Para testar e usar o validador de solicitação no console, siga as instruções em [Testar a validação básica de solicitações usando o console do API Gateway \(p. 352\)](#).

Testar a validação básica de solicitações no API Gateway

Escolha um dos tópicos a seguir para obter instruções sobre como testar a validação básica de solicitações com a [API de amostra \(p. 353\)](#).

Tópicos

- [Testar a validação básica de solicitações usando a API REST do API Gateway \(p. 350\)](#)
- [Testar a validação básica de solicitações usando o console do API Gateway \(p. 352\)](#)

Testar a validação básica de solicitações usando a API REST do API Gateway

Para ver a URL de invocação da API implantada, você pode exportar a API do estágio, certificando-se de incluir o cabeçalho `Accept: application/json` ou `Accept: application/yaml`:

```
GET /restapis/fjd6crafxc/stages/testStage/exports/swagger?extensions=validators HTTP/1.1
Accept: application/json
Content-Type: application/json
Host: apigateway.us-east-1.amazonaws.com
X-Amz-Date: 20170306T234936Z
Authorization: AWS4-HMAC-SHA256 Credential={access_key_ID}/20170306/us-east-1/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature={sig4_hash}
```

É possível ignorar o parâmetro de consulta `?extensions=validators` quando você não deseja fazer download das especificações do OpenAPI relacionadas à validação de solicitações.

Para testar a validação de solicitações usando as chamadas da API REST do API Gateway

1. Chame GET `/validation?q1=cat`.

```
GET /testStage/validation?q1=cat HTTP/1.1
Host: fjd6crafxc.execute-api.us-east-1.amazonaws.com
Content-Type: application/json
Accept: application/json
```

Como o parâmetro necessário de `q1` está definido e não está em branco, a solicitação passa na validação. O API Gateway retorna a seguinte resposta 200 OK:

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

2. Chame GET `/validation`.

```
GET /testStage/validation HTTP/1.1
Host: fjd6crafxc.beta.execute-api.us-east-1.amazonaws.com
Content-Type: application/json
Accept: application/json
```

Como o parâmetro necessário de q1 não está definido, a solicitação não consegue passar na validação. O API Gateway retorna a seguinte resposta 400 Bad Request:

```
{  
    "message": "Missing required request parameters: [q1]"  
}
```

3. Chame GET /validation?q1=.

```
GET /testStage/validation?q1= HTTP/1.1  
Host: fjd6crafxc.execute-api.us-east-1.amazonaws.com  
Content-Type: application/json  
Accept: application/json
```

Como o parâmetro necessário de q1 está em branco, a solicitação não consegue passar na validação. O API Gateway retorna a mesma resposta 400 Bad Request que no exemplo anterior.

4. Chame POST /validation.

```
POST /testStage/validation HTTP/1.1  
Host: fjd6crafxc.execute-api.us-east-1.amazonaws.com  
Content-Type: application/json  
Accept: application/json  
h1: v1  
  
{  
    "name" : "Marco",  
    "type" : "dog",  
    "price" : 260  
}
```

Como o parâmetro necessário de h1 está definido e não está em branco, e o formato da carga atende às propriedades necessárias e às restrições associadas de RequestDataModel, a solicitação passará na validação. O API Gateway retorna a seguinte resposta bem-sucedida.

```
{  
    "pet": {  
        "name": "Marco",  
        "type": "dog",  
        "price": 260  
    },  
    "message": "success"  
}
```

5. Chame POST /validation sem especificar o cabeçalho h1 ou definir seu valor em branco.

```
POST /testStage/validation HTTP/1.1  
Host: fjd6crafxc.execute-api.us-east-1.amazonaws.com  
Content-Type: application/json  
Accept: application/json  
  
{  
    "name" : "Marco",  
    "type" : "dog",  
    "price" : 260  
}
```

Como o parâmetro de cabeçalho necessário de h1 está ausente ou definido como em branco, a solicitação não consegue passar na validação. O API Gateway retorna a seguinte resposta 400 Bad Request:

```
{  
    "message": "Missing required request parameters: [h1]"  
}
```

6. Chame POST /validation, definindo a propriedade type da carga como bird.

```
POST /testStage/validation HTTP/1.1  
Host: fjd6crafxc.execute-api.us-east-1.amazonaws.com  
Content-Type: application/json  
Accept: application/json  
X-Amz-Date: 20170309T000215Z  
h1: v1  
  
{  
    "name" : "Molly",  
    "type" : "bird",  
    "price" : 269  
}
```

Como o valor da propriedade type não é um membro de enumeração de ["dog", "cat", "fish"], a solicitação não consegue passar na validação. O API Gateway retorna a seguinte resposta 400 Bad Request:

```
{  
    "message": "Invalid request body"  
}
```

Configurar price como 501 viola a restrição na propriedade. Isso faz com que a validação falhe e a mesma resposta 400 Bad Request seja retornada.

Testar a validação básica de solicitações usando o console do API Gateway

As etapas a seguir descrevem como testar a validação básica de solicitações no console do API Gateway.

Para testar a validação de solicitações em um método usando TestInvoke no console do API Gateway

Enquanto estiver conectado ao console do API Gateway, faça o seguinte:

1. Escolha Resources para a API para a qual você configurou um mapa de validadores de solicitação.
2. Escolha um método para o qual você habilitou a validação de solicitações com um validador de solicitação especificado.
3. Em Method Execution, na caixa Client, escolha Test.
4. Experimente diferentes valores para o parâmetro de solicitação necessário ou para o corpo aplicável e escolha Test para ver a resposta.

Quando a chamada de método passar na validação, você obterá as respostas esperadas. Se a validação falhar, a seguinte mensagem de erro será retornada se a carga não estiver no formato correto:

```
{  
    "message": "Invalid request body"
```

```
}
```

Se os parâmetros de solicitação não forem válidos, a seguinte mensagem de erro será retornada:

```
{
    "message": "Missing required request parameters: [p1]"
}
```

Definições do OpenAPI de uma API de amostra com a validação básica de solicitações

A seguinte definição do OpenAPI 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.

Existem dois validadores de solicitação declarados no mapa `x-amazon-apigateway-request-validators` em nível de API. O validador `params-only` está habilitado na API e é herdado pelo método `GET`. Esse validador permite que o API Gateway verifique se o parâmetro de consulta necessário (`q1`) está incluído e não está em branco na solicitação de entrada. O validador `all` está habilitado no método `POST`. Esse validador verifica se o parâmetro de cabeçalho necessário (`h1`) está definido e não está em branco. Ele também verifica se o formato da carga atende ao esquema `RequestBodyModel` especificado. Esse modelo 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.

Para obter mais informações sobre o comportamento dessa API, consulte [Ativar a validação de solicitação no API Gateway \(p. 343\)](#).

OpenAPI 2.0

```
{
    "swagger": "2.0",
    "info": {
        "title": "ReqValidators Sample",
        "version": "1.0.0"
    },
    "schemes": [
        "https"
    ],
    "basePath": "/v1",
    "produces": [
        "application/json"
    ],
    "x-amazon-apigateway-requestValidators" : {
        "all" : {
            "validateRequestBody" : true,
            "validateRequestParameters" : true
        },
        "params-only" : {
            "validateRequestBody" : false,
            "validateRequestParameters" : true
        }
    },
    "x-amazon-apigateway-request-validator" : "params-only",
    "paths": {
        "/validation": {
            "post": {
                "x-amazon-apigateway-request-validator" : "all",
                "parameters": [
                    ...
                ]
            }
        }
    }
}
```

```
{  
    "in": "header",  
    "name": "h1",  
    "required": true  
},  
{  
    "in": "body",  
    "name": "RequestBodyModel",  
    "required": true,  
    "schema": {  
        "$ref": "#/definitions/RequestBodyModel"  
    }  
}  
],  
"responses": {  
    "200": {  
        "schema": {  
            "type": "array",  
            "items": {  
                "$ref": "#/definitions/Error"  
            }  
        },  
        "headers" : {  
            "test-method-response-header" : {  
                "type" : "string"  
            }  
        }  
    },  
    "security" : [{  
        "api_key" : []  
    }],  
    "x-amazon-apigateway-auth" : {  
        "type" : "none"  
    },  
    "x-amazon-apigateway-integration" : {  
        "type" : "http",  
        "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",  
        "httpMethod" : "POST",  
        "requestParameters": {  
            "integration.request.header.custom_h1": "method.request.header.h1"  
        },  
        "responses" : {  
            "2\\d{2}" : {  
                "statusCode" : "200"  
            },  
            "default" : {  
                "statusCode" : "400",  
                "responseParameters" : {  
                    "method.response.header.test-method-response-header" : "'static value'"  
                },  
                "responseTemplates" : {  
                    "application/json" : "json 400 response template",  
                    "application/xml" : "xml 400 response template"  
                }  
            }  
        }  
    },  
    "get": {  
        "parameters": [  
            {  
                "name": "q1",  
                "in": "query",  
                "required": true  
            }  
        ]  
    }  
}
```

```
        ],
        "responses": {
            "200": {
                "schema": {
                    "type": "array",
                    "items": {
                        "$ref": "#/definitions/Error"
                    }
                },
                "headers" : {
                    "test-method-response-header" : {
                        "type" : "string"
                    }
                }
            },
            "security" : [
                "api_key" : []
            ],
            "x-amazon-apigateway-auth" : {
                "type" : "none"
            },
            "x-amazon-apigateway-integration" : {
                "type" : "http",
                "uri" : "http://petstore-demo-endpoint.execute-api.com/petstore/pets",
                "httpMethod" : "GET",
                "requestParameters": {
                    "integration.request.querystring.type": "method.request.querystring.q1"
                },
                "responses" : {
                    "2\\d{2}" : {
                        "statusCode" : "200"
                    },
                    "default" : {
                        "statusCode" : "400",
                        "responseParameters" : {
                            "method.response.header.test-method-response-header" : "'static value'"
                        },
                        "responseTemplates" : {
                            "application/json" : "json 400 response template",
                            "application/xml" : "xml 400 response template"
                        }
                    }
                }
            }
        },
        "definitions": {
            "RequestBodyModel": {
                "type": "object",
                "properties": {
                    "id": { "type": "integer" },
                    "type": { "type": "string", "enum": ["dog", "cat", "fish"] },
                    "name": { "type": "string" },
                    "price": { "type": "number", "minimum": 25, "maximum": 500 }
                },
                "required": [ "type", "name", "price" ]
            },
            "Error": {
                "type": "object",
                "properties": {
                }
            }
        }
    }
```

}

Importar uma API REST no API Gateway

Você pode usar o recurso Import API do API Gateway para importar uma API REST de um arquivo de definição externo para o API Gateway. Atualmente, o recurso Import API oferece suporte para arquivos de definição do [OpenAPI v2.0](#) e [OpenAPI v3.0](#). 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` na URL solicitada.

Para obter um tutorial sobre o uso do recurso Import API no console do API Gateway, consulte [TUTORIAL: Crie uma API REST importando um exemplo \(p. 45\)](#).

Tópicos

- [Importar uma API otimizada para fronteiras para o API Gateway \(p. 356\)](#)
- [Importar uma API regional para o API Gateway \(p. 357\)](#)
- [Importar um arquivo do OpenAPI para atualizar uma definição de API existente \(p. 357\)](#)
- [Definir a propriedade `basePath` do OpenAPI \(p. 359\)](#)
- [Erros e avisos durante a importação \(p. 361\)](#)

Importar uma API otimizada para fronteiras 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. Você pode 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: Crie uma API REST importando um exemplo \(p. 45\)](#).

Tópicos

- [Importar uma API otimizada para fronteiras usando o console do API Gateway \(p. 356\)](#)
- [Importar uma API otimizada para fronteiras usando a AWS CLI \(p. 356\)](#)

Importar uma API otimizada para fronteiras usando o console do API Gateway

Para importar uma API otimizada para fronteiras usando o console do API Gateway, faça o seguinte:

1. Faça login no console do API Gateway e escolha + Create API (+ Criar API).
2. Selecione a opção Import from OpenAPI (Importar do OpenAPI) em Create new API (Criar nova API).
3. Copie uma definição do OpenAPI da API e cole-a no editor de código ou escolha Select OpenAPI File (Selecionar arquivo do OpenAPI) para carregar um arquivo do OpenAPI de uma unidade local.
4. Em Settings (Configurações), para Endpoint Type (Tipo de endpoint), escolha Edge optimized.
5. Escolha Import (Importar) para começar a importar as definições do OpenAPI.

Importar uma API otimizada para fronteiras usando a AWS CLI

Para importar uma API de um arquivo de definição do OpenAPI a fim de criar uma nova API otimizada para fronteiras usando a AWS CLI, use o comando `import-rest-api` da seguinte forma:

```
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 \
--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. Use o console do API Gateway, a AWS CLI ou o AWS SDK.

Ao exportar uma API, a configuração de endpoint da API não está incluída nas definições da API exportada.

Para obter um tutorial sobre o uso do recurso Import API no console do API Gateway, consulte [TUTORIAL: Crie uma API REST importando um exemplo \(p. 45\)](#).

Tópicos

- [Importar uma API regional usando o console do API Gateway \(p. 357\)](#)
- [Importar uma API regional usando a AWS CLI \(p. 357\)](#)

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. Faça login no console do API Gateway e escolha Criar API.
2. Selecione a opção Import from OpenAPI (Importar do OpenAPI) em Create new API (Criar nova API).
3. Copie uma definição do OpenAPI da API e cole-a no editor de código ou escolha Select OpenAPI File (Selecionar arquivo do OpenAPI) para carregar um arquivo do OpenAPI de uma unidade local.
4. Em Settings (Configurações), para Endpoint Type (Tipo de endpoint), escolha Regional.
5. Escolha Import (Importar) para começar a importar as definições do OpenAPI.

Importar uma API regional usando a AWS CLI

Para importar uma API de um arquivo de definição do OpenAPI usando a AWS CLI, use o comando `import-rest-api`:

```
aws apigateway import-rest-api \
--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 for mesclada com B, a API resultante produzirá os seguintes métodos:

```
GET /a
```

que retornará as respostas 200 e 400 e o método

```
POST /a
```

que retornará as respostas 200 e 206.

A mesclagem de uma API é útil quando você decompondo 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 (p. 112)

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 (p. 112)

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 colocará a API no mesmo estado final, como se a primeira operação tivesse sido bem-sucedida.

Definir a propriedade `basePath` do 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 faça referência ao seu estágio de produção.

Note

O API Gateway criará 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 criará recursos intermédios automaticamente para você, mesmo que eles não estejam explicitamente declarados na sua definição.

dividir

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.

Erros e avisos 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.

Controlar e gerenciar acesso a uma API REST no API Gateway

O API Gateway oferece suporte a vários mecanismos de controle 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” \(p. 362\)](#).
- As funções e políticas padrão do AWS IAM oferecem controles de acesso flexíveis e robustos que podem ser aplicados a toda uma API ou métodos individuais. As funções e políticas do IAM podem ser usadas para controlar quem pode criar e gerenciar suas APIs, bem como quem pode invocá-las. Para obter mais informações, consulte [the section called “Usar permissões do IAM” \(p. 378\)](#).
- As tags do IAM podem ser usadas em conjunto com as políticas do IAM para controlar o acesso. Para obter mais informações, consulte [the section called “Controle de acesso baseado em tags” \(p. 720\)](#).
- Políticas de endpoint para VPC endpoints de interface permitem anexar políticas de recurso do IAM a VPC endpoints 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” \(p. 394\)](#).

- Autorizadores do Lambda são funções Lambda que controlam o acesso aos métodos da API REST usando a autenticação de token de portador, bem como as informações descritas pelos cabeçalhos, caminhos, strings de consulta, variáveis de estágio ou parâmetros de solicitação de variáveis de contexto. Os autorizadores do Lambda são usados para controlar quem pode invocar métodos da API REST. Para obter mais informações, consulte [the section called “Usar autorizadores do Lambda” \(p. 396\)](#).
- Os grupos de usuários do Amazon Cognito permitem que você crie as soluções personalizáveis de autenticação e autorização 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 Cognito como um autorizador para uma API REST” \(p. 414\)](#).

Os mecanismos a seguir podem ser usados 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 “Habilitar o CORS para um recurso da API REST” \(p. 423\)](#).
- Os certificados SSL no lado do cliente podem ser usados para verificar se as solicitações HTTP para seu sistema de back-end provêm do API Gateway. Para obter mais informações, consulte [the section called “Gerar e configurar um certificado SSL para autenticação de back-end” \(p. 429\)](#).
- O AWS WAF pode ser usado para proteger sua API do API Gateway contra explorações comuns da web. Para obter mais informações, consulte [the section called “Use o AWS WAF para proteger sua API contra explorações comuns da web” \(p. 449\)](#).

Os mecanismos a seguir podem ser usados para rastrear e limitar o acesso que você concedeu aos 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 “Utilização de planos de uso com chaves de API” \(p. 450\)](#).

Controlar o acesso a uma API usando as políticas de recursos do Amazon API Gateway

As políticas de recursos do Amazon API Gateway são documentos de política JSON que você anexa a uma API para controlar se um principal especificado (geralmente um usuário ou uma função 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 determinada origem
- nuvens privadas virtuais (VPCs) ou VPC endpoints (em qualquer conta) específicos

Você pode usar as políticas de recursos para todos os tipos de endpoint da API no API Gateway: privado, regional e otimizado para fronteiras.

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” \(p. 394\)](#).

Você pode anexar uma política de recursos a uma API usando o console da AWS, a CLI da AWS ou SDKs da AWS.

As políticas de recursos do API Gateway são diferentes das políticas do IAM. As políticas do IAM são anexadas a entidades do IAM (usuários, grupos ou funções) e definem quais ações essas entidades são capazes de realizar em cada recurso. As políticas de recursos do API Gateway são anexadas aos recursos. Para uma discussão mais detalhada sobre as diferenças entre políticas baseadas em identidade (IAM) e políticas de recursos, consulte [Políticas baseadas em identidade e em recursos](#).

Você pode usar as políticas de recursos do API Gateway junto com as políticas do IAM.

Tópicos

- [Visão geral da linguagem de políticas de acesso para o Amazon API Gateway \(p. 363\)](#)
- [Como as políticas de recursos do Amazon API Gateway afetam o fluxo de trabalho de autorização. \(p. 364\)](#)
- [Exemplos de política de recursos do API Gateway \(p. 373\)](#)
- [Criar e anexar uma política de recursos do API Gateway para uma API \(p. 375\)](#)
- [Chaves de condição da AWS que podem ser usadas nas políticas de recursos do API Gateway \(p. 377\)](#)

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 informações completas sobre linguagem de políticas, consulte [Visão geral de políticas do IAM](#) e [Referência de política 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. Para o formato do elemento `Resource`, consulte [Formato de recurso das permissões para executar a API no API Gateway \(p. 387\)](#).
- Ações – para cada recurso, o Amazon API Gateway oferece suporte a um conjunto de operações. Você identifica as operações de recurso que permitirá (ou negará) usando palavras-chave de ação.

Por exemplo, a `apigateway: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 \(p. 386\)](#).

- Efeito – qual será o efeito quando o usuário solicitar 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

"Implicit deny" é a mesma coisa que "Deny by default".

"Implicit deny" é diferente de "Explicit deny". Para obter mais informações, consulte [A diferença entre negação por padrão e negação explícita](#).

- Principal – a conta ou usuário que tem permissão de acesso a ações e recursos na declaração. Em uma política de recursos, o principal é o usuário ou a conta do IAM que serão destinatários dessa 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.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": "*",  
            "Action": "execute-api:Invoke",  
            "Resource": "arn:aws:execute-api:region:account-id:*"  
        },  
        {  
            "Effect": "Deny",  
            "Principal": "*",  
            "Action": "execute-api:Invoke",  
            "Resource": "arn:aws:execute-api:region:account-id:*",  
            "Condition": {  
                "NotIpAddress": {  
                    "aws:SourceIp": "123.4.5.6/24"  
                }  
            }  
        }  
    ]  
}
```

Como as políticas de recursos do Amazon 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.

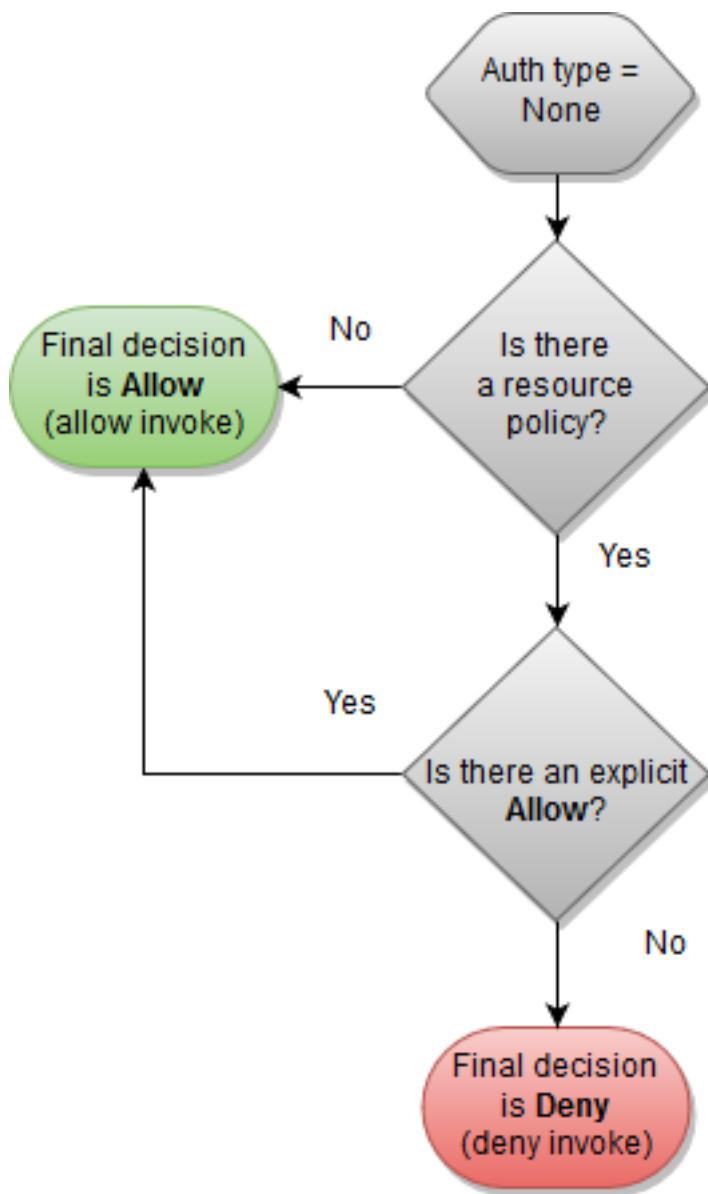
Tópicos

- [Somente política de recursos do API Gateway \(p. 364\)](#)
- [Política de recursos e autorizador do Lambda \(p. 366\)](#)
- [Política de recursos e autenticação do IAM \(p. 368\)](#)
- [Política de recursos e autenticação do Amazon Cognito \(p. 370\)](#)
- [Tabelas de resultados de avaliação de política \(p. 372\)](#)

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 envolverá a busca de uma permissão explícita, com base nos critérios de entrada do chamador. Uma negação implícita ou qualquer negação explícita resultará na negação do chamador.

Veja a seguir um exemplo dessa política de recursos.

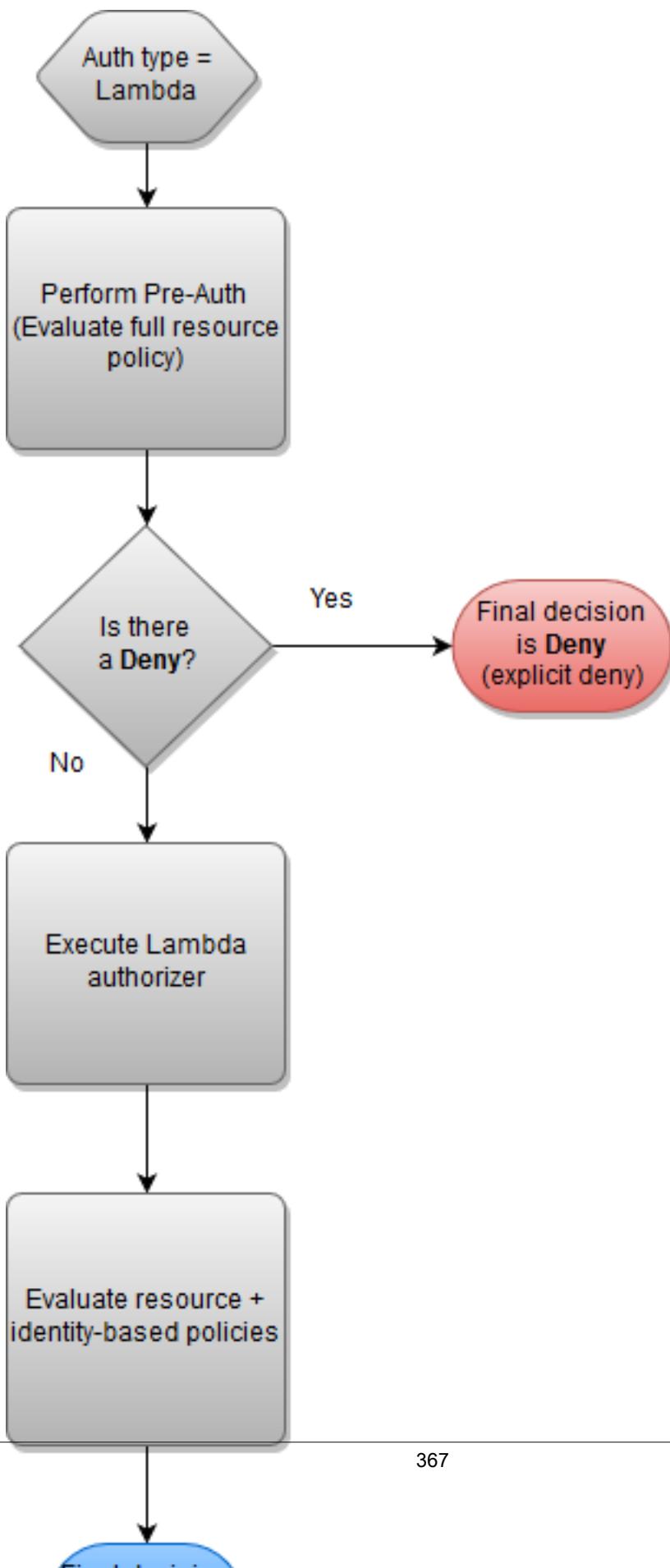


```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": "*",  
            "Action": "execute-api:Invoke",  
            "Resource": "arn:aws:execute-api:<region>:<account-id>:<api-id>/",  
            "Condition": {  
                "IpAddress": {  
                    "aws:SourceIp": ["192.0.2.0/24", "198.51.100.0/24"]  
                }  
            }  
        }  
    ]  
}
```

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 será avaliada em duas fases. Antes de chamar o autorizador do Lambda, primeiro o API Gateway avaliará a política e verificará se existem negações explícitas. Se encontradas, o chamador terá o acesso negado imediatamente. Caso contrário, o autorizador do Lambda é chamado e retorna um [documento de política \(p. 408\)](#), que é avaliado em conjunto com a política de recursos. O resultado é determinado com base na [Tabela A \(p. 372\)](#) abaixo.

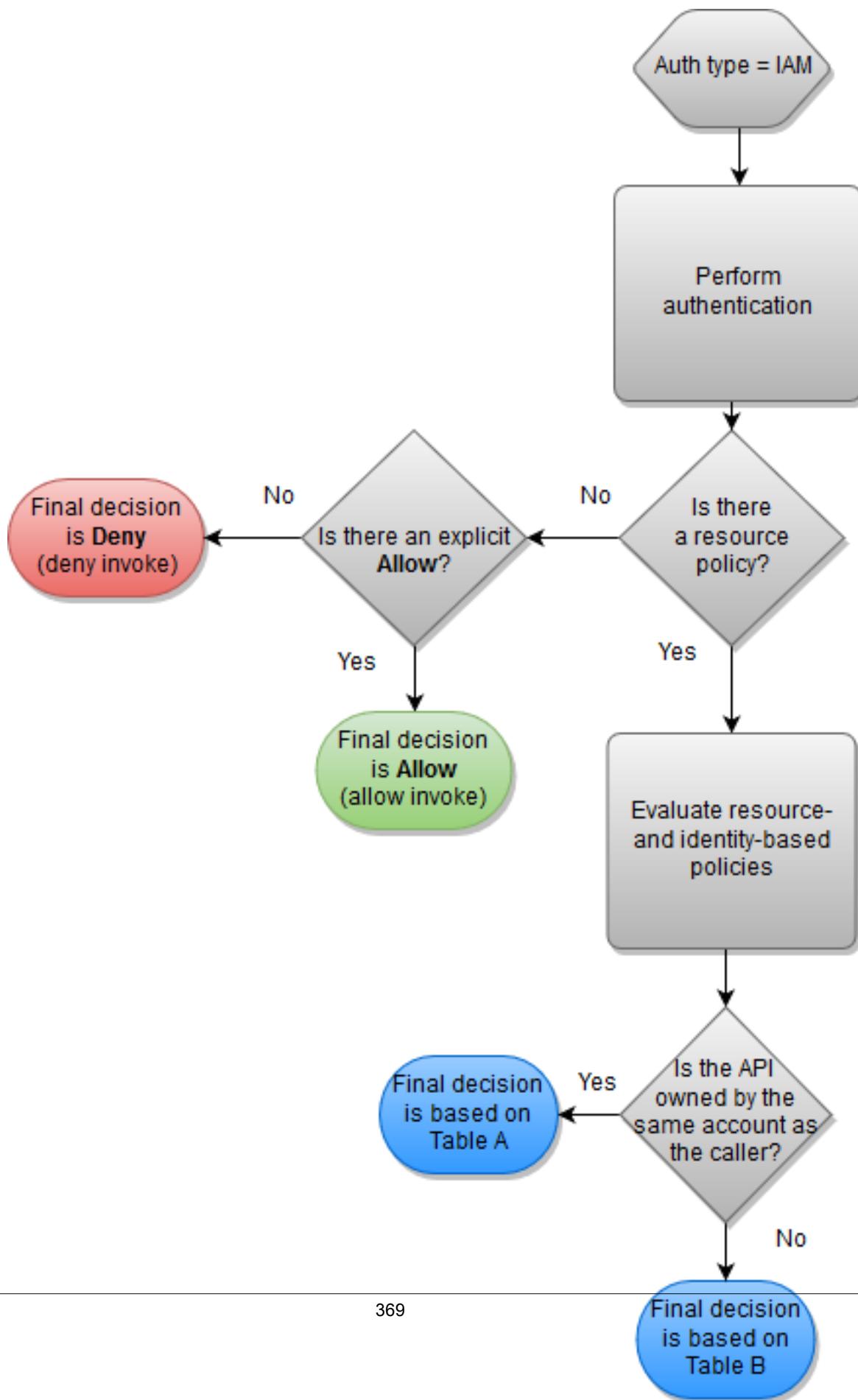
O exemplo de política de recursos a seguir permite chamadas somente a partir do VPC endpoint cujo ID de VPC endpoint é `vpce-1a2b3c4d`. Durante a avaliação de pré-autorização, somente as chamadas vindas do VPC endpoint indicado abaixo serão permitidas para prosseguir e avaliar o autorizador do Lambda. Todas as chamadas restantes serão bloqueadas.



```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Deny",  
            "Principal": "*",  
            "Action": "execute-api:Invoke",  
            "Resource": [  
                "arn:aws:execute-api:region:account-id:api-id/*"  
            ],  
            "Condition": {  
                "StringNotEquals": {  
                    "aws:SourceVpce": "vpce-1a2b3c4d"  
                }  
            }  
        }  
    ]  
}
```

Política de recursos e autenticação do IAM

Neste fluxo de trabalho, uma autenticação do IAM é configurada para a API além de uma política de recursos. Após autenticar o usuário com o serviço do IAM, as políticas anexadas ao usuário do IAM, além da política de recursos, são avaliadas em conjunto. O resultado variará com base na origem do chamador, se ele está na mesma conta do proprietário da API ou em outra conta da AWS. Se o chamador e o proprietário da API forem de contas diferentes, as políticas de usuário do IAM e a política de recursos devem permitir explicitamente que o chamador prossiga. (Consulte a [Tabela B \(p. 372\)](#) abaixo.) Em contraste, se o chamador e o proprietário da API estiverem na mesma conta, as políticas de usuário ou a política de recursos deve permitir explicitamente que o chamador prossiga. (Consulte a [Tabela A \(p. 372\)](#) abaixo.)

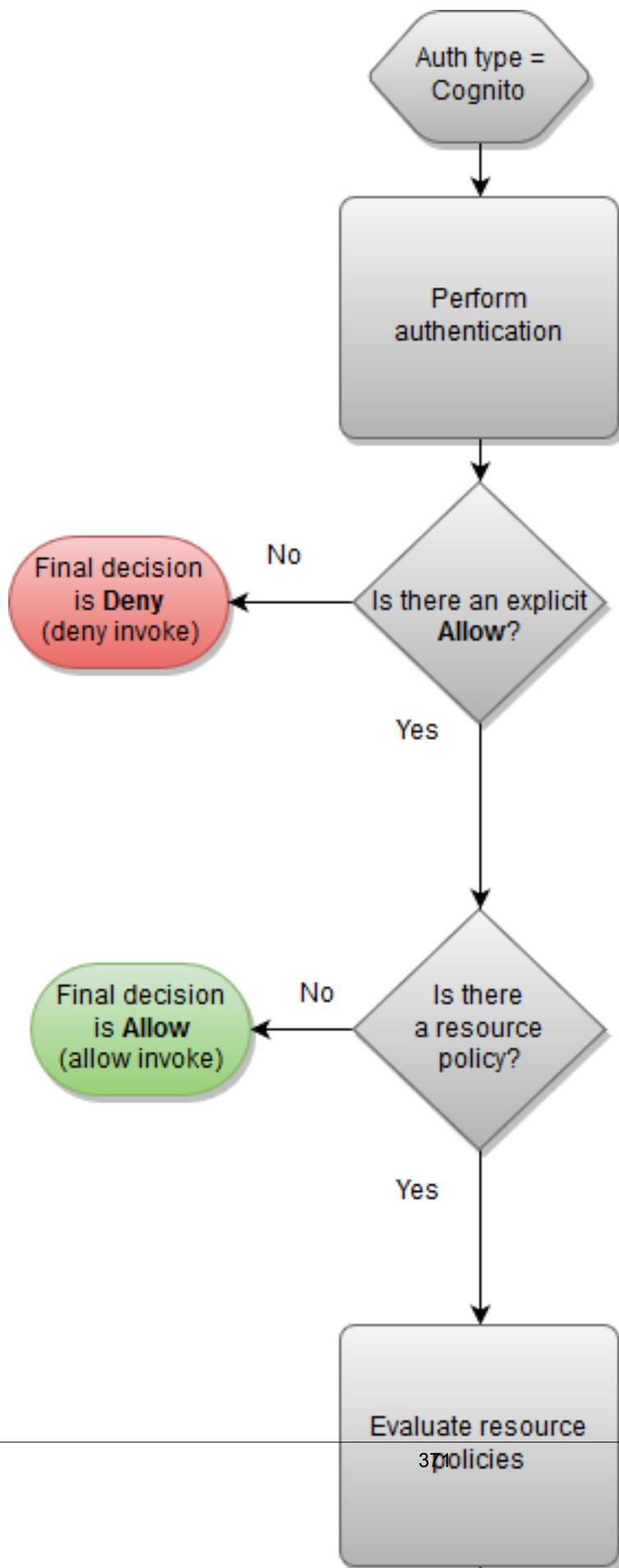


Veja a seguir um exemplo de uma política de recursos entre contas. Supondo que a política de usuário do IAM contém uma Permissão, essa política de recursos permitirá chamadas somente da VPC cujo ID de VPC é [vpc-2f09a348](#). (Consulte a [Tabela B \(p. 372\)](#) abaixo.)

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": "",  
            "Action": "execute-api:Invoke",  
            "Resource": [  
                "arn:aws:execute-api:region:account-id:api-id/*"  
            ],  
            "Condition" : {  
                "StringEquals": {  
                    "aws:SourceVpc": "vpc-2f09a348"  
                }  
            }  
        }  
    ]  
}
```

Política de recursos e autenticação do Amazon Cognito

Neste fluxo de trabalho, um [grupo de usuários do Amazon Cognito \(p. 414\)](#) é configurado para a API além da política de recursos. Primeiro o API Gateway tenta autenticar o chamador por meio do Amazon Cognito. Isso normalmente é realizado por meio de um [token JWT](#) fornecido pelo chamador. 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, resultará em uma negação. Veja a seguir um exemplo de uma política de recursos que pode ser usada junto com o Grupos de usuários do Amazon Cognito.



Veja a seguir um exemplo de uma política de recursos que permite chamadas somente a partir de IPs de origem especificados, assumindo que o token de autenticação do Amazon Cognito contém uma Permissão. Consulte a [Tabela A \(p. 372\)](#) abaixo.)

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": "*",  
            "Action": "execute-api:Invoke",  
            "Resource": "arn:aws:execute-api:<region>:<account-id>:<api-id>/",  
            "Condition": {  
                "IpAddress": {  
                    "aws:SourceIp": ["192.0.2.0/24", "198.51.100.0/24"]  
                }  
            }  
        }  
    ]  
}
```

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 ou Grupos de usuários do Amazon Cognito) e uma política de recursos do API Gateway, ambas na mesma conta da AWS.

Tabela A: a conta A chama a API pertencente à conta A

Política de usuário do IAM (ou autorizador do Lambda ou Grupos de usuários do Amazon Cognito)	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
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 do Lambda ou Grupos de usuários do Amazon Cognito) e uma política de recursos do API Gateway, que estão em diferentes contas da AWS. Se uma delas for silenciosa (nem permissão nem negação), o acesso entre contas é negado. Isso acontece pois o acesso entre contas requer que tanto a política de recursos quanto a política do IAM (ou um autorizador do Lambda ou Grupos de usuários do Amazon Cognito) conceda acesso explícito.

Tabela B: a conta B chama a API pertencente à conta A

Política de usuário do IAM (ou autorizador do Lambda ou 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
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. As políticas usam as sequências `account-id` e `api-id` no valor do recurso. Para testar essas políticas, você precisará substituir essas strings pelo seu próprio ID de conta e ID de API. Para obter informações sobre a linguagem de políticas de acesso, consulte [Visão geral da linguagem de políticas de acesso para o Amazon API Gateway \(p. 363\)](#).

Tópicos

- [Exemplo: permitir que usuários em outra conta da AWS usem uma API \(p. 373\)](#)
- [Exemplo: negar tráfego da API com base no intervalo ou endereço IP de origem \(p. 374\)](#)
- [Exemplo: permitir tráfego da API privada com base na VPC ou no VPC endpoint de origem \(p. 374\)](#)

Exemplo: permitir que usuários em outra conta da AWS usem uma API

Este exemplo de política de recursos concede acesso à API em uma conta da AWS para dois usuários em uma outra conta da AWS por meio de protocolos [Signature Version 4](#) (SigV4). Especificamente, Alice e o usuário raiz da conta da AWS identificada por `account-id-2` recebem acesso à ação `execute-api:Invoke` para executar a ação `GET` no recurso `pets` (API) na conta da AWS identificada por `account-id-1`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::account-id-2:user/Alice",
                    "account-id-2"
                ]
            },
            "Action": "execute-api:Invoke",
            "Resource": "arn:aws:apigateway:us-east-1:execute-api::/restapis/api-id/resources/resource-id/methods/httpMethod"
        }
    ]
}
```

```
        "Resource": [
            "arn:aws:execute-api:region:account-id-1:api-id/stage/GET/pets"
        ]
    }
}
```

Exemplo: negar tráfego da API com base no intervalo ou endereço IP de origem

Este exemplo de política de recursos é uma política de "lista negra" que nega (bloqueia) o tráfego de entrada a uma API de dois blocos de endereços IP de origem especificados.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "arn:aws:execute-api:region:account-id:api-id/*"
            ]
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "arn:aws:execute-api:region:account-id:api-id/*"
            ],
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": ["192.0.2.0/24", "198.51.100.0/24"]
                }
            }
        }
    ]
}
```

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:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "arn:aws:execute-api:region:account-id:api-id/*"
            ]
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "arn:aws:execute-api:region:account-id:api-id/*"
            ],
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": ["192.0.2.0/24", "198.51.100.0/24"]
                }
            }
        }
    ]
}
```

```
"Resource": [
    "arn:aws:execute-api:region:account-id:api-id/*"
],
"Condition" : {
    "StringNotEquals": {
        "aws:SourceVpc": "vpc-1a2b3c4d"
    }
}
]
```

Este exemplo de política de recurso especifica um VPC endpoint de origem:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "arn:aws:execute-api:region:account-id:api-id/*"
            ]
        },
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "execute-api:Invoke",
            "Resource": [
                "arn:aws:execute-api:region:account-id:api-id/*"
            ],
            "Condition" : {
                "StringNotEquals": {
                    "aws:SourceVpce": "vpce-1a2b3c4d"
                }
            }
        }
    ]
}
```

Criar e anexar uma política de recursos do API Gateway para uma API

Para permitir que um usuário acesse sua API chamando o serviço de execução da API, você precisa criar uma política de recursos do API Gateway (que controla o acesso aos recursos do API Gateway) e anexar a política à API.

Important

Para atualizar uma política de recurso do API Gateway, você precisará ter a permissão `apigateway:UpdateRestApiPolicy`, além da permissão `apigateway:PATCH`.

A política de recursos pode ser anexada à API, quando a API está sendo criada, ou pode ser anexada posteriormente. Em relação às APIs privadas, observe que até que você anexe a política de recursos para a API privada, todas as chamadas de API falharão.

Important

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 [Implantando uma API REST no Amazon API Gateway \(p. 516\)](#).

O acesso pode ser controlado por meio de elementos de condição do IAM, incluindo condições na conta da AWS, VPC de origem, VPC endpoint de origem ou intervalo de IP. Se o elemento `Principal` na política estiver definido como `*`, outros tipos de autorização poderão ser usados junto com a política de recurso. Se o elemento `Principal` estiver definido como `AWS`, haverá uma falha na autorização para todos os recursos não protegidos com a autorização `AWS_IAM`, incluindo recursos não protegidos.

As seções a seguir descrevem como criar sua própria política de recursos do API Gateway e anexá-la à sua API. A anexação de uma política aplica as permissões da política aos métodos na API.

Important

Se você usar o console do API Gateway para anexar uma política de recursos a uma API implantada ou se você atualizar uma política de recursos existente, será necessário implantar a API no console novamente para que as alterações entrem em vigor.

Tópicos

- [Anexar políticas de recursos do API Gateway \(console\) \(p. 376\)](#)
- [Anexar políticas de recursos do API Gateway \(CLI da AWS\) \(p. 376\)](#)

Anexar políticas de recursos do API Gateway (console)

Você pode usar o Console de Gerenciamento da AWS para anexar uma política de recursos a uma API do API Gateway.

Para anexar uma política de recursos a uma API do API Gateway

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha o nome da API.
3. No painel de navegação à esquerda, escolha Política de recursos.
4. Se você quiser, escolha um dos Examples (Exemplos). 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.

Se você não usar um dos Examples (Exemplos), insira a política de recurso.

5. Escolha Salvar.

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.

Anexar políticas de recursos do API Gateway (CLI da AWS)

Para usar a AWS CLI para criar uma nova API e anexar uma política de recursos a ela, chame o comando `create-rest-api` desta forma:

```
aws apigateway create-rest-api \
--name "api-name" \
--policy "{\"jsonEscapedPolicyDocument\"}"
```

Para usar a AWS CLI para anexar uma política de recursos a uma API existente, chame o comando `update-rest-api` desta forma:

```
aws apigateway update-rest-api \
```

```
--rest-api-id api-id \  
--patch-operations op=replace,path=/policy,value='{"jsonEscapedPolicyDocument"}'
```

Chaves de condição da AWS que podem ser usadas nas políticas de recursos do API Gateway

A tabela a seguir contém a lista completa das 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 globais de contexto de condição da AWS](#).

Tabela de chaves de condição

Chaves de condição	Critérios	Precisa de AuthN ?	Tipo de autorização
<code>aws:CurrentTime</code>	Nenhum	Não	Tudo
<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:MultiFactorAuth</code>	A chave está presente somente em solicitações assinadas com credenciais de segurança temporárias.	Sim	IAM
<code>aws:MultiFactorAuth</code>	A chave estará presente somente se o MFA estiver presente nas solicitações.	Sim	IAM
<code>aws:PrincipalType</code>	Nenhum	Sim	IAM
<code>aws:Referer</code>	A chave estará presente somente se o valor for fornecido pelo chamador 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
<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

Chaves de condição	Critérios	Precisa de AuthN ?	Tipo de autorização
<code>aws:UserAgent</code>	A chave estará presente somente se o valor for fornecido pelo chamador 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 com permissões do IAM

Você controla o acesso à API do Amazon API Gateway com [permissões IAM](#), controlando o acesso aos dois processos de componentes 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 chamador de 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 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 do IAM](#) que representa o desenvolvedor, a um [grupo do IAM](#) que contém o usuário ou a uma função IAM assumida pelo usuário.

Neste documento de políticas do IAM, o elemento `Resource` do IAM contém uma lista de entidades de API do API Gateway, incluindo [recursos do API Gateway](#) e [relações de links do API Gateway](#). O elemento `Action` do IAM contém as ações necessárias de gerenciamento de API do API Gateway. Essas ações são declaradas no formato `apigateway:HTTP_VERB`, em que `apigateway` designa o componente de gerenciamento de APIs subjacente do API Gateway e `HTTP_VERB` representa os verbos HTTP com suporte pelo API Gateway.

Para obter mais informações sobre como usar esse modelo de permissões, consulte [Controlar o acesso para gerenciar uma API \(p. 380\)](#).

Modelo de permissões do API Gateway para invocar uma API

Para permitir que um chamador de API invoque a API ou atualize seu armazenamento em cache, você deve criar políticas IAM que permitam que esse invoque o método de API para o qual a autenticação de usuários do IAM está habilitada. O desenvolvedor de APIs define a propriedade `authorizationType` do método como `AWS_IAM` para exigir que o chamador envie as chaves de acesso do usuário do IAM a serem autenticadas. Em seguida, você anexa a política a um usuário do IAM que representa o chamador de API, a um grupo do IAM que contém esse usuário ou a uma função IAM assumida pelo usuário.

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: InvalidateCache`, 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 \(p. 384\)](#).

Quando uma API está integrada a um serviço da AWS (por exemplo, o AWS Lambda) no back-end, o API Gateway também deve ter permissões para acessar os recursos integrados da AWS (por exemplo, invocar uma função Lambda) em nome do chamador da API. Para conceder essas permissões, crie uma função IAM do tipo AWS service for 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:

```
{  
    "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` do 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 de SDK correspondente.

Para que o API Gateway chame o serviço da AWS integrado, você também deve anexar a essa função as políticas de permissão adequadas do IAM para chamar serviços integrados da AWS. Por exemplo, para chamar uma função Lambda, inclua a seguinte política de permissão do IAM na função do IAM:

```
{  
    "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 Lambda usando o console do API Gateway, você não é solicitado a definir essa função IAM explicitamente, pois o console define as permissões com base em recurso na função 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 no chamador, no qual uma política de permissões é diretamente anexada ao usuário ou grupo do IAM do agente de chamada, ou o modelo de permissões com base em função, no qual uma política de permissões é anexada a uma função 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 tirar proveito disso para exigir que um usuário do IAM acesse um serviço da AWS somente por meio da API do API Gateway.

Controlar o acesso para gerenciar uma API

Nesta seção, você aprenderá a criar instruções de políticas do IAM para controlar quem pode ou não criar, implantar e atualizar uma API no API Gateway. Você também encontrará a referência a instruções de política, incluindo os formatos dos campos `Action` e `Resource` relacionados ao serviço de gerenciamento de API.

Controlar quem pode criar e gerenciar uma API do API Gateway com políticas IAM

Para controlar quem pode ou não criar, implantar e atualizar sua API usando o serviço de gerenciamento de API do API Gateway, crie um documento de política IAM com as permissões necessárias, conforme indicado no modelo de política a seguir:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Permission",  
            "Action": [  
                "apigateway:HTTP_VERB"  
            ],  
            "Resource": [  
                "arn:aws:apigateway:region::resource1-path",  
                "arn:aws:apigateway:region::resource2-path",  
                ...  
            ]  
        }  
    ]  
}
```

Aqui, **Permission** pode ser `Allow` ou `Deny`, para conceder ou revogar, respectivamente, os direitos de acesso, conforme estipulado pela instrução de política. Para obter mais informações, consulte [Permissões do IAM na AWS](#).

HTTP_VERB pode ser qualquer um dos verbos HTTP com suporte pelo [API Gateway \(p. 381\)](#). * pode ser usado para representar qualquer um dos verbos HTTP.

Resource contém uma lista de ARNs das entidades de API afetadas, incluindo `RestApi`, `Resource`, `Method`, `Integration`, `DocumentationPart`, `Model`, `Authorizer`, `UsagePlan`, etc. Para obter mais informações, consulte [Formato de recurso das permissões para gerenciar a API no API Gateway \(p. 382\)](#).

Combinando diferentes instruções de política diferentes, você pode personalizar as permissões de acesso para usuários individuais, grupos ou funções para acessar entidades de API selecionadas e realizar ações especificadas de acordo com essas entidades. Por exemplo, você pode incluir a seguinte instrução na política do IAM para conceder à sua equipe de documentação a permissão para criar, publicar, atualizar e excluir [partes de documentação](#) de uma API especificada, bem como para visualizar as entidades de API.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "apigateway:GET"  
            ],  
            "Resource": [  
                "arn:aws:apigateway:region::restapi_id:resource1-path",  
                "arn:aws:apigateway:region::restapi_id:resource2-path",  
                ...  
            ]  
        }  
    ]  
}
```

```
        ],
        "Resource": [
            "arn:aws:apigateway:<region>::/restapis/<api-id>/*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "apigateway:POST",
            "apigateway:PATCH",
            "apigateway:DELETE"
        ],
        "Resource": [
            "arn:aws:apigateway:<region>::/restapis/<api-id>/documentation/*"
        ]
    }
]
```

Para a sua equipe de desenvolvimento principal da API que é responsável por todas as operações, você pode incluir a seguinte instrução na política do IAM para conceder a ela permissões de acesso muito mais amplas.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "apigateway:*"
            ],
            "Resource": [
                "arn:aws:apigateway:*>::/*"
            ]
        }
    ]
}
```

Referência de instrução de políticas IAM para gerenciar a API no API Gateway

As informações a seguir descrevem o formato de elementos Action e Resource usado em uma instrução de política IAM para conceder ou revogar permissões para gerenciar entidades de API do API Gateway.

Formato de ação das permissões para gerenciar a API no API Gateway

A expressão Action de gerenciamento de API possui o seguinte formato geral:

```
apigateway:<action>
```

em que **action** é uma das seguintes ações do API Gateway:

- *, que representa todas as ações a seguir.
- GET, que é usado para obter informações sobre recursos.
- POST, que é usado principalmente para criar recursos filho.
- PUT, que é usado principalmente para atualizar recursos (e, embora não seja recomendado, pode ser usado para criar recursos filho).
- DELETE, que é usado para excluir recursos.
- PATCH, que pode ser usado para atualizar recursos.

Alguns exemplos da expressão Action incluem:

- **apigateway:*** para todas as ações do API Gateway.
- **apigateway:GET** para apenas a ação GET no API Gateway.

Formato de recurso das permissões para gerenciar a API no API Gateway

A expressão Resource de gerenciamento de API possui o seguinte formato geral:

```
arn:aws:apigateway:region:::resource-path-specifier
```

em que **region** é uma região de destino da AWS (como **us-east-1** ou ***** para todas as regiões da AWS com suporte) e **resource-path-specifier** é o caminho para os recursos de destino.

Algumas expressões de recursos de exemplo incluem:

- **arn:aws:apigateway:**region**::/restapis/*** para todos os recursos, métodos, modelos e estágios na região da AWS **region**.
- **arn:aws:apigateway:**region**::/restapis/**api-id**/*** para todos os recursos, métodos, modelos e estágios na API com o identificador **api-id** na região da AWS **region**.
- **arn:aws:apigateway:**region**::/restapis/**api-id**/resources/**resource-id**/*** para todos os recursos e métodos no recurso com o identificador **resource-id**, que está na API com o identificador **api-id** na região da AWS **region**.
- **arn:aws:apigateway:**region**::/restapis/**api-id**/resources/**resource-id**/methods/*** para todos os métodos no recurso com o identificador **resource-id**, que está na API com o identificador **api-id** na região da AWS **region**.
- **arn:aws:apigateway:**region**::/restapis/**api-id**/resources/**resource-id**/methods/GET** para apenas o método GET no recurso com o identificador **resource-id**, que está na API com o identificador **api-id** na região da AWS **region**.
- **arn:aws:apigateway:**region**::/restapis/**api-id**/models/*** para todos os modelos na API com o identificador **api-id** na região da AWS **region**.
- **arn:aws:apigateway:**region**::/restapis/**api-id**/models/**model-name**** para o modelo com nome **model-name**, que está na API com o identificador **api-id** na região da AWS **region**.
- **arn:aws:apigateway:**region**::/restapis/**api-id**/stages/*** para todos os estágios na API com o identificador **api-id** na região da AWS **region**.
- **arn:aws:apigateway:**region**::/restapis/**api-id**/stages/**stage-name**** apenas para o estágio com o nome **stage-name**, na API com o identificador **api-id** na região da AWS **region**.

Controlar acesso entre contas à sua API

Você pode gerenciar o acesso às suas APIs criando políticas de permissão do IAM para controlar quem pode ou não criar, atualizar, implantar, visualizar ou excluir entidades da API. Uma política é anexada a um usuário do IAM que representa seu usuário, a um grupo do IAM que contém o usuário ou a uma função do IAM assumida pelo usuário.

Nas políticas do IAM que você cria para as suas APIs, você pode usar elementos Condition para permitir o acesso somente a determinadas integrações do Lambda ou autorizadores.

O bloco Condition usa operadores de condição booleana para corresponder à condição na política com valores na solicitação. O operador de condição String**Xxx** funcionará tanto para integração da AWS (em que o valor deve ser o ARN de uma função do Lambda) quanto para integração HTTP (em que o valor deve ser um URI HTTP). Os seguintes operadores de condição String**Xxx** são compatíveis: **StringEquals**, **StringNotEquals**, **StringEqualsIgnoreCase**, **StringNotEqualsIgnoreCase**,

`StringLike`, `StringNotLike`. Para mais informações, consulte [Operadores de condição de cadeia](#) no Guia do usuário do IAM.

Política do IAM para autorizador do Lambda entre contas

Veja a seguir o exemplo de uma política do IAM para controlar uma função de autorizador do Lambda entre contas:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "apigateway:POST"  
            ],  
            "Resource": [  
                "arn:aws:apigateway:[region]::/restapis/restapi_id/authorizers"  
            ],  
            //Create Authorizer operation is allowed only with the following Lambda  
            function  
            "Condition": {  
                "StringEquals": {  
                    "apigateway:AuthorizerUri":  
                        "arn:aws:apigateway:region:lambda:path/2015-03-31/functions/  
                        arn:aws:lambda:region:123456789012:function:example/invocations"  
                }  
            }  
        }  
    ]  
}
```

Política do IAM para integração do Lambda entre contas

Com a integração entre contas, para restringir as operações em alguns recursos específicos (como `put-integration` para uma função Lambda específica), um elemento `Condition` pode ser adicionado à política para especificar qual recurso (função Lambda) será afetado.

Veja a seguir o exemplo de uma política do IAM para controlar uma função de integração do Lambda entre contas:

Para conceder a outra conta da AWS permissão para chamar `integration:put` ou `put-integration` e configurar uma integração do Lambda na sua API, você pode incluir a seguinte declaração na política do IAM.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "apigateway:PUT"  
            ],  
            "Resource": [  
                "arn:aws:apigateway:api-region::/restApis/api-id/resources/resource-id/  
                methods/GET/integration"  
            ],  
            //PutIntegration is only valid with the following Lambda function  
            "Condition": {  
                "StringEquals": {  
                    "apigateway:IntegrationUri": "arn:aws:lambda:region:account-  
                    id:function:lambda-function-name"  
                }  
            }  
        }  
    ]  
}
```

```
        }
    ]
}
```

Permitir que outra conta gerencie a função do Lambda usada ao importar um arquivo do OpenAPI

Para conceder a outra conta da AWS a permissão para chamar `restapi:import` ou `import-rest-api` para importar um arquivo do OpenAPI, você pode incluir a seguinte instrução na política do IAM.

Na instrução Condition abaixo, a string "lambda:path/2015-03-31/functions/
arn:aws:lambda:**us-east-1:account-id**:function:**lambda-function-name**" é o ARN
completo para a função Lambda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "apigateway:POST"
            ],
            "Resource": "arn:aws:apigateway:*/:::restapis",
            "Condition": {
                "StringLike": {
                    "apigateway:IntegrationUri": [
                        "arn:aws:apigateway:apigateway-region:lambda:path/2015-03-31/
functions/arn:aws:lambda:lambda-region:account-id:function:lambda-function-name/
invocations"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "apigateway:POST"
            ],
            "Resource": "arn:aws:apigateway:*/:::restapis",
            "Condition": {
                "StringLike": {
                    "apigateway:AuthorizerUri": [
                        "arn:aws:apigateway:apigateway-region:lambda:path/2015-03-31/
functions/arn:aws:lambda:lambda-region:account-id:function:lambda-function-name/
invocations"
                    ]
                }
            }
        }
    ]
}
```

Controlar o acesso para chamar uma API

Nesta seção, você aprenderá a escrever instruções de política do IAM para controlar quem pode chamar uma API implantada no API Gateway. Aqui, você também encontrará a referência da instrução de política, incluindo os formatos dos campos Action e Resource relacionados ao serviço de execução da API. Você também deve examinar a seção do IAM em [the section called “Como as políticas de recursos afetam o fluxo de trabalho de autorização” \(p. 364\)](#).

Para APIs privadas, é necessário usar uma combinação de uma política de VPC endpoint e uma política de recurso do API Gateway. Para obter mais informações, consulte os tópicos a seguir:

- the section called “[Usar políticas de recursos do API Gateway](#)” (p. 362)
- the section called “[Usar políticas de VPC endpoint para APIs privadas](#)” (p. 394)

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.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Permission",  
            "Action": [  
                "execute-api:Execution-operation"  
            ],  
            "Resource": [  
                "arn:aws:execute-api:region:account-id:api-id/stage/METHOD_HTTP_VERB/Resource-path"  
            ]  
        }  
    ]  
}
```

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 IAM para executar a API no API Gateway](#) (p. 386).

Note

Para que as políticas 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:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "execute-api:Invoke"  
            ],  
            "Resource": [  
                "arn:aws:execute-api:us-east-1:account-id:api-id/*/GET/pets"  
            ]  
        },  
        {  
            "Effect": "Deny",  
            "Action": [  
                "execute-api:Invoke"  
            ],  
            "Resource": [  
                "arn:aws:execute-api:us-east-1:account-id:api-id/*/PUT/pets"  
            ]  
        }  
    ]  
}
```

```
    "Action": [
        "execute-api:Invoke"
    ],
    "Resource": [
        "arn:aws:execute-api:us-east-1:account-id: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:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "execute-api:Invoke"
            ],
            "Resource": [
                "arn:aws:execute-api:us-east-1:account-id:api-id/*/GET/pets/a1b2"
            ]
        }
    ]
}
```

Para as APIs de teste de uma equipe de desenvolvedores, você pode incluir a seguinte instrução na política do IAM para permitir que a equipe chame qualquer método em qualquer recurso de qualquer API por qualquer desenvolvedor no estágio test.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "execute-api:Invoke",
                "execute-api: InvalidateCache"
            ],
            "Resource": [
                "arn:aws:execute-api:***:*/test/*"
            ]
        }
    ]
}
```

Referência de instrução de políticas IAM para executar a API no API Gateway

As informações a seguir descrevem o formato de Ação e Recurso das instruções de política 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.
- Invocar, 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. (* pode ser usado para todas as APIs, independentemente do identificador da API.)
- **stage-name** é o nome do estágio associado ao método (* pode ser usado para todos os estágios, independentemente do nome do estágio.)
- **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. (* pode ser usado para todos os caminhos.)

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. (Isso é equivalente a *).
- **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 **api-id** na região da AWS us-east-1.
- **arn:aws:execute-api:us-east-1::*:**api-id**/test/*** para o caminho do recurso no estágio de test, para a API com o identificador **api-id** na região da AWS us-east-1.
- **arn:aws:execute-api:us-east-1::*:**api-id**/test/*/**mydemoresource**/*** para qualquer caminho de recurso ao longo do caminho **mydemoresource**, para qualquer método HTTP no estágio test, para a API com o identificador **api-id** na região da AWS us-east-1.
- **arn:aws:execute-api:us-east-1::*:**api-id**/test/GET/**mydemoresource**/*** para métodos GET em qualquer caminho de recurso ao longo do caminho **mydemoresource**, no estágio test, para a API com o identificador **api-id** na região da AWS us-east-1.

Exemplos de políticas do IAM para o gerenciamento de APIs do API Gateway

Os documentos de política de exemplo a seguir mostram vários casos de uso para definir permissões de acesso para o gerenciamento de recursos de API no API Gateway. Para o modelo de permissões e outras informações de segundo plano, consulte [Controlar quem pode criar e gerenciar uma API do API Gateway com políticas IAM \(p. 380\)](#).

Tópicos

- [Permissões de leitura simples \(p. 388\)](#)

- Permissões somente leitura em qualquer API (p. 388)
- Permissões de acesso completo a qualquer recurso do API Gateway (p. 389)
- Permissões de acesso completo para o gerenciamento de estágios de API (p. 390)
- Impedir que usuários especificados excluam quaisquer recursos de API (p. 390)

Permissões de leitura simples

A seguinte instrução de política dá ao usuário a permissão para obter informações sobre todos os recursos, métodos, modelos e estágios na API com o identificador a123456789 na região da AWS us-east-1:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "apigateway:GET"  
            ],  
            "Resource": [  
                "arn:aws:apigateway:us-east-1:::/restapis/a123456789/*"  
            ]  
        }  
    ]  
}
```

A instrução de política do exemplo a seguir dá ao usuário do IAM a permissão para listar informações para todos os recursos, métodos, modelos e etapas em qualquer região. O usuário também tem permissão para realizar todas as ações disponíveis do API Gateway para a API com o identificador a123456789 na região da AWS us-east-1:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "apigateway:GET"  
            ],  
            "Resource": [  
                "arn:aws:apigateway:*/::/restapis/*"  
            ]  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "apigateway:/*"  
            ],  
            "Resource": [  
                "arn:aws:apigateway:us-east-1:::/restapis/a123456789/*"  
            ]  
        }  
    ]  
}
```

Permissões somente leitura em qualquer API

O seguinte documento de política permitirá que entidades anexadas (usuários, grupos ou funções) recuperem qualquer uma das APIs da conta da AWS do agente de chamada. Isso inclui qualquer um dos recursos filho de uma API, como método, integração etc.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "Stmt1467321237000",  
            "Effect": "Deny",  
            "Action": [  
                "apigateway:POST",  
                "apigateway:PUT",  
                "apigateway:PATCH",  
                "apigateway:DELETE"  
            ],  
            "Resource": [  
                "arn:aws:apigateway:us-east-1::/*"  
            ]  
        },  
        {  
            "Sid": "Stmt1467321341000",  
            "Effect": "Deny",  
            "Action": [  
                "apigateway:GET"  
            ],  
            "Resource": [  
                "arn:aws:apigateway:us-east-1::/",  
                "arn:aws:apigateway:us-east-1::/account",  
                "arn:aws:apigateway:us-east-1::/clientcertificates",  
                "arn:aws:apigateway:us-east-1::/domainnames",  
                "arn:aws:apigateway:us-east-1::/apikeys"  
            ]  
        },  
        {  
            "Sid": "Stmt1467321344000",  
            "Effect": "Allow",  
            "Action": [  
                "apigateway:GET"  
            ],  
            "Resource": [  
                "arn:aws:apigateway:us-east-1::/restapis/*"  
            ]  
        }  
    ]  
}
```

A primeira instrução Deny proíbe explicitamente quaisquer chamadas de POST, PUT, PATCH e DELETE em qualquer recurso do API Gateway. Isso garante que essas permissões não serão substituídas por outros documentos de política também anexados ao agente de chamada. A segunda instrução Deny bloqueia o agente de chamada para consultar o recurso raiz (/), informações da conta (/account), certificados de cliente (/clientcertificates), nomes de domínio personalizados (/domainnames) e chaves de API (/apikeys). Juntas, as três declarações garantem que o chamador só possa consultar recursos relacionados à API. Isso pode ser útil em testes de API quando você não deseja que o testador modifique nenhum código.

Para restringir o acesso somente leitura acima a APIs especificadas, substitua a propriedade Resource da instrução Allow pelo seguinte:

```
"Resource": ["arn:aws:apigateway:us-east-1::/restapis/restapi_id1/*",  
            "arn:aws:apigateway:us-east-1::/restapis/restapi_id2/*"]
```

Permissões de acesso completo a qualquer recurso do API Gateway

O documento de política de exemplo a seguir concede acesso completo a qualquer um dos recursos do API Gateway da conta da AWS.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "Stmt1467321765000",  
            "Effect": "Allow",  
            "Action": [  
                "apigateway:*"  
            ],  
            "Resource": [  
                "*"  
            ]  
        }  
    ]  
}
```

Em geral, você deve evitar o uso de uma política de acesso ampla e aberta. Pode ser necessário fazer isso para a sua equipe principal de desenvolvimento de APIs, para que os membros possam criar, implantar, atualizar e excluir qualquer recurso do API Gateway.

Permissões de acesso completo para o gerenciamento de estágios de API

Os seguintes documentos de política concedem permissões de acesso total em recursos relacionados a Stage de qualquer API na conta da AWS do agente de chamada.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "apigateway:*"  
            ],  
            "Resource": [  
                "arn:aws:apigateway:us-east-1:::/restapis/*/stages",  
                "arn:aws:apigateway:us-east-1:::/restapis/*/*/  
            ]  
        }  
    ]  
}
```

O documento de política acima concede permissões de acesso completo somente à coleção stages e a qualquer um dos recursos stage contidos, desde que nenhuma outra política que conceda mais acessos tenha sido anexada ao agente de chamada. Caso contrário, você deverá negar explicitamente todos os acessos.

Usando a política acima, o agente de chamada deve descobrir com o identificador da API REST antecedência, pois o usuário não pode chamar GET /restapis para consultar as APIs disponíveis. Além disso, se arn:aws:apigateway:us-east-1:::/restapis/*/*/
Resource, o recurso Stages se tornará inacessível. Nesse caso, o agente de chamada não poderá criar um estágio nem obter os estágios existentes, embora ele ainda possa visualizar, atualizar ou excluir um estágio, desde que o nome desse estágio seja conhecido.

Para conceder permissões para os estágios de uma API específica, basta substituir a parte restapis/* das especificações Resource por restapis/*restapi_id*, em que *restapi_id* é o identificador da API de interesse.

Impedir que usuários especificados excluam quaisquer recursos de API

O seguinte documento de política IAM de exemplo impede que um usuário especificado exclua qualquer recurso de API no API Gateway.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "Stmt1467331998000",  
            "Effect": "Allow",  
            "Action": [  
                "apigateway:GET",  
                "apigateway:PATCH",  
                "apigateway:POST",  
                "apigateway:PUT"  
            ],  
            "Resource": [  
                "arn:aws:apigateway:us-east-1:::/restapis/*"  
            ]  
        },  
        {  
            "Sid": "Stmt1467332141000",  
            "Effect": "Allow",  
            "Action": [  
                "apigateway:DELETE"  
            ],  
            "Condition": {  
                "StringNotLike": {  
                    "aws:username": "johndoe"  
                }  
            },  
            "Resource": [  
                "arn:aws:apigateway:us-east-1:::/restapis/*"  
            ]  
        }  
    ]  
}
```

Esta política IAM concede permissão de acesso completo para criar, implantar, atualizar e excluir uma API para usuários, grupos ou funções anexados, exceto para o usuário especificado (`johndoe`), que não pode excluir recursos de API. Ela pressupõe que nenhum outro documento de política concedendo permissões `Allow` na raiz, chaves de API, certificados de cliente ou nomes de domínio personalizados tenha sido anexado ao agente de chamada.

Para impedir que o usuário especificado exclua recursos específicos do API Gateway, por exemplo, uma API específica ou os recursos de uma API, substitua a especificação `Resource` acima por isto:

```
"Resource": ["arn:aws:apigateway:us-east-1:::/restapis/restapi_id_1",  
            "arn:aws:apigateway:us-east-1:::/restapis/restapi_id_2/resources"]
```

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 \(p. 384\)](#).

A seguinte instrução de política fornece ao usuário permissão para chamar qualquer método POST ao longo do caminho `mydemoresource` no estágio `test`, para a API com o identificador `a123456789`, supondo que a API correspondente tenha sido implantada na região da AWS us-east-1:

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

```
{  
    "Effect": "Allow",  
    "Action": [  
        "execute-api:Invoke"  
    ],  
    "Resource": [  
        "arn:aws:execute-api:us-east-1:*:a123456789/test/POST/mydemoresource/*"  
    ]  
}  
}
```

O exemplo de instrução de política a seguir concede ao usuário a permissão para chamar qualquer método no caminho de recurso de petstorewalkthrough/pets, em qualquer fase, para a API com o identificador de a123456789, em qualquer região da AWS em que a API correspondente tenha sido implantada:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "execute-api:Invoke"  
            ],  
            "Resource": [  
                "arn:aws:execute-api:***:a123456789/*/*/petstorewalkthrough/pets"  
            ]  
        }  
    ]  
}
```

Criar e anexar uma política para um usuário do IAM

Para permitir que um usuário chame o serviço de gerenciamento de API ou o serviço de execução de API, você deve criar uma política IAM para um usuário do IAM, que controle o acesso a entidades do API Gateway, e, em seguida, anexar a política ao usuário do IAM. As etapas a seguir descrevem como criar sua política IAM.

Para criar sua própria política IAM

1. Faça login no Console de gerenciamento da AWS e abra o console da IAM em <https://console.aws.amazon.com/iam/>.
2. Selecione Policies (Políticas) e depois Create Policy (Criar política). Se aparecer um botão Get Started, selecione-o e, em seguida, Create Policy.
3. Próximo a Create Your Own Policy, escolha Select.
4. Em Nome da política, digite qualquer valor que seja mais fácil para você consultar mais tarde. Você também pode digitar um texto descritivo em Descrição.
5. Em Documento de política, digite uma instrução com o seguinte formato e escolha Criar política:

```
{  
    "Version": "2012-10-17",  
    "Statement" : [  
        {  
            "Effect" : "Allow",  
            "Action" : [  
                "action-statement"  
            ],  
            "Resource": "  
                "arn:aws:execute-api:us-east-1:  
                    a123456789/test/  
                    POST/  
                    mydemoresource/*"  
            ]  
        }  
    ]  
}
```

```
    "Resource" : [
        "resource-statement"
    ],
},
{
    "Effect" : "Allow",
    "Action" : [
        "action-statement"
    ],
    "Resource" : [
        "resource-statement"
    ]
}
}
```

Nesta instrução, substitua **action-statement** e **resource-statement** conforme necessário e adicione outras instruções para especificar a entidades do API Gateway que o usuário do IAM terá permissão para gerenciar, os métodos de API que o usuário do IAM pode chamar ou ambos. Por padrão, o usuário do IAM não possui permissões, a menos que haja uma declaração Allow explícita correspondente.

6. Para habilitar a política para um usuário, escolha Usuários.
7. Escolha o usuário do IAM para o qual você deseja anexar a política.

Você acabou de criar uma política IAM. Ela não terá efeito até que você a anexe a um usuário do IAM, a um grupo IAM que contém esse usuário ou a uma função IAM assumida por esse usuário.

Para anexar uma política IAM a um usuário do IAM

1. Para o usuário escolhido, selecione a guia Permissões e escolha Anexar política.
2. Em Conceder permissões, escolha Anexar políticas existentes diretamente.
3. Escolha o documento de política que você acabou de criar na lista exibida e escolha Próximo: Revisar.
4. Em Resumo de permissões, escolha Adicionar permissões.

Como alternativa, você pode adicionar o usuário a um grupo do IAM, caso ele ainda não seja membro, e anexar o documento de política ao grupo para que as políticas anexadas sejam aplicáveis a todos os membros do grupo. É útil gerenciar e atualizar as configurações de políticas em um grupo de usuários do IAM. A seguir, destacamos como anexar a política a um grupo do IAM, supondo que você já tenha criado o grupo e adicionado o usuário a ele.

Para anexar um documento de política IAM a um grupo do IAM

1. No painel de navegação principal, escolha Grupos.
2. Escolha a guia Permissões no grupo escolhido.
3. Escolha Anexar política.
4. Escolha o documento de política que você criou anteriormente e depois selecione Anexar política.

Para o API Gateway chamar outros serviços da AWS em seu nome, crie uma função IAM do tipo Amazon API Gateway.

Para criar um tipo de função do Amazon API Gateway

1. No painel de navegação principal, escolha Funções.
2. Escolha Criar nova função.

3. Digite um nome para Nome da função e escolha Próxima etapa.
4. Em Selecionar tipo de função, Funções de serviço da AWS, escolha Selecionar ao lado de Amazon API Gateway.
5. Escolha uma política gerenciada disponível de permissões do IAM, por exemplo, AmazonAPIGatewayPushToCloudWatchLog se você deseja que o API Gateway registre métricas no CloudWatch, em Anexar política e depois escolha Próxima etapa.
6. Em Entidades confiáveis, verifique se apigateway.amazonaws.com está listado como uma entrada e escolha Criar função.
7. Na função recém-criada, selecione a guia Permissões e escolha Anexar política.
8. Escolha o documento de política personalizada do IAM criada anteriormente e depois selecione Anexar política.

Usar políticas de VPC endpoint para APIs privadas no API Gateway

É possível melhorar a segurança das [APIs privadas](#) configurando o API Gateway para usar um [VPC endpoint de interface](#). 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. Para obter mais informações sobre a criação de VPC endpoints, consulte [Criação de um endpoint de interface](#).

Uma política de VPC endpoint é uma política de recurso do IAM que pode ser anexada a um VPC endpoint de interface para controlar o acesso ao endpoint. Para obter mais informações, consulte [Controlar o acesso a serviços com VPC endpoints](#). É possível usar uma política de endpoint para restringir o tráfego que vai da sua rede interna para acessar as APIs privadas. É possível optar por permitir ou não o acesso a APIs privadas específicas que podem ser acessadas por meio do VPC endpoint.

As políticas de VPC endpoint podem ser usadas em conjunto com políticas de recurso do API Gateway. A política de recurso é usada para especificar quais principais podem acessar a API. A política de endpoint especifica quais APIs privadas podem ser chamadas pelo VPC endpoint. Para obter mais informações sobre políticas de recursos, consulte [the section called “Usar políticas de recursos do API Gateway” \(p. 362\)](#).

Exemplos de política de VPC endpoint

Você pode criar políticas de Amazon Virtual Private Cloud endpoints para o Amazon API Gateway nas quais é possível especificar:

- O principal que pode executar ações.
- As ações que podem ser executadas.
- Os recursos que podem ter ações executadas neles.

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 VPC endpoints](#).

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.

```
{
```

```
"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.

```
{
    "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.

```
{
    "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/*"
            ]
        }
    ]
}
```

}

Usar tags para controlar o acesso a uma API REST no API Gateway

A permissão para acessar APIs REST pode ser ajustada usando o controle de acesso baseado em tags nas políticas do IAM.

Para obter mais informações, consulte [the section called “Controle de acesso baseado em tags” \(p. 720\)](#).

Usar autorizadores do Lambda do API Gateway

Um autorizador do Lambda (anteriormente conhecido como autorizador personalizado) é um recurso do API Gateway que usa uma função do Lambda para controlar o acesso à sua API.

Um autorizador do Lambda é útil se você deseja implementar um esquema de autorização personalizado que usa uma estratégia de autenticação de token de portador, como OAuth ou SAML, ou que usa parâmetros de solicitação para determinar a identidade do chamador.

Quando um cliente faz uma solicitação para um dos métodos da sua API, o API Gateway chama o autorizador do Lambda, que assume a identidade do chamador como uma entrada e retorna uma política do IAM como uma saída.

Há dois tipos de autorizadores do Lambda:

- Um autorizador baseado em token do Lambda (também chamado de autorizador `TOKEN`) recebe a identidade do chamador em um token de portador, como token de web JSON (JWT) ou um token OAuth.
- Um autorizador de solicitação baseada em parâmetro do Lambda (também chamado de autorizador `REQUEST`) recebe a identidade do chamador em uma combinação de cabeçalhos, parâmetros de strings de consulta, [stageVariables \(p. 315\)](#) e variáveis `$context (p. 306)`.

Para APIs WebSocket, apenas autorizadores de solicitação baseados em parâmetro são compatíveis.

É possível usar uma função AWS Lambda de uma conta da AWS que é diferente daquela na qual você criou sua função Lambda de autorizador. Para obter mais informações, consulte [the section called “Configurar um autorizador do Lambda entre contas” \(p. 412\)](#).

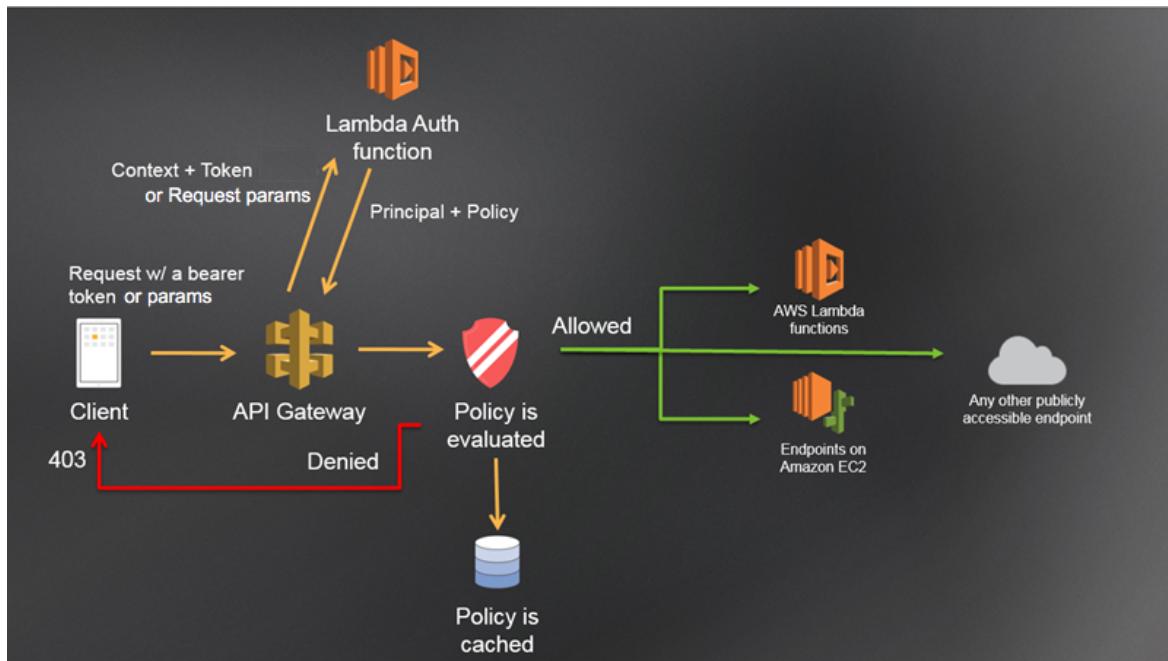
Para ver um exemplo de um autorizador .NET do Lambda, visualize o vídeo [Criar um autorizador personalizado do API Gateway no .NET Core 2.0](#).

Tópicos

- [Fluxo de trabalho do autorizador do Lambda \(p. 397\)](#)
- [Etapas para criar um autorizador do Lambda do API Gateway \(p. 397\)](#)
- [Criar uma função do autorizador do Lambda do API Gateway no console do Lambda \(p. 398\)](#)
- [Configurar um autorizador do Lambda usando o console do API Gateway \(p. 403\)](#)
- [Configurar um autorizador do Lambda usando a AWS CLI ou um AWS SDK \(p. 405\)](#)
- [Entrada para um autorizador do Lambda do Amazon API Gateway \(p. 406\)](#)
- [Saída de um autorizador do Lambda do Amazon API Gateway \(p. 408\)](#)
- [Chamar uma API com autorizadores do Lambda do API Gateway \(p. 409\)](#)
- [Configurar um autorizador do Lambda entre contas \(p. 412\)](#)

Fluxo de trabalho do autorizador do Lambda

O diagrama a seguir ilustra o fluxo de autorização para autorizadores do Lambda.



Fluxo de trabalho de autorização do Lambda do API Gateway

1. O cliente chama um método em um método de API do API Gateway, repassando um token de portador ou parâmetros de solicitação.
2. O API Gateway verifica se um autorizador do Lambda está configurado para o método. Se estiver, o API Gateway chamará a função do Lambda.
3. A função do Lambda autentica o chamador por meios como os seguintes:
 - Chamada para um provedor OAuth para obter um token de acesso OAuth.
 - Chamada para um provedor SAML para obter uma declaração do SAML.
 - Geração de uma política do IAM com base nos valores de parâmetro de solicitação.
 - Recuperação de credenciais de um banco de dados.
4. Se a chamada for bem-sucedida, a função do Lambda concederá acesso retornando um objeto de saída que contém, pelo menos, um identificador principal e uma política do IAM.
5. O API Gateway avalia a política.
 - 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 executará o método. Se o armazenamento em cache estiver habilitado nas configurações do autorizador, o API Gateway também armazenará em cache a política para que a função do autorizador do Lambda não precise ser invocada novamente.

Etapas para criar um autorizador do Lambda do API Gateway

Para criar o autorizador do Lambda, você precisa executar as seguintes tarefas:

1. Crie a função do autorizador do Lambda no console do Lambda, conforme descrito em [the section called “Criar uma função do autorizador do Lambda no console do Lambda” \(p. 398\)](#). Você pode

- usar um dos exemplos de esquema como ponto de partida e personalizar a [entrada \(p. 406\)](#) e a [saída \(p. 408\)](#) conforme desejado.
2. Configure a função do Lambda como um autorizador do API Gateway e configure um método de API para exigir-la, conforme descrito em [the section called “Configurar um autorizador do Lambda usando o console” \(p. 403\)](#). Como alternativa, se você precisa de um autorizador do Lambda entre contas, consulte [the section called “Configurar um autorizador do Lambda entre contas” \(p. 412\)](#).

Note

Também é possível configurar um autorizador usando a AWS CLI ou um AWS SDK. Consulte [the section called “Configurar um autorizador do Lambda usando a AWS CLI ou um AWS SDK” \(p. 405\)](#).

3. Teste seu autorizador usando [Postman](#) conforme descrito em [the section called “Chamar uma API com autorizadores do Lambda” \(p. 409\)](#).

Criar uma função do autorizador do Lambda do API Gateway no console do Lambda

Antes de configurar um autorizador do Lambda, você deve criar a função do Lambda que implementa a lógica para autorizar e, se necessário, para autenticar o chamador. 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. Caso contrário, você precisará usar um dos esquemas no repositório GitHub [awslabs](#) como ponto de partida.

Para o exemplo de funções do autorizador do Lambda nesta seção, que não chamam outros serviços, você pode usar o [AWSLambdaBasicExecutionRole](#) integrado. Ao criar a função do Lambda para seu autorizador do Lambda do API Gateway, você precisará atribuir uma função de execução do IAM à função do Lambda se ela chamar outros serviços da AWS. Para criar a função, siga as instruções na [Função de execução AWS Lambda](#).

EXEMPLO: Criar uma função do autorizador do Lambda com base em token

Para criar uma função Lambda de autorizador com base em token, insira o seguinte código Node.js 8.10 no console do Lambda e teste-o no console do API Gateway da seguinte forma:

1. No console do Lambda, escolha Create function (Criar função).
2. Escolha Author from scratch.
3. Insira um nome para a função.
4. Em Runtime (Tempo de execução), selecione Node.js 8.10.
5. Selecione Create function (Criar função).
6. Copie/cole o código a seguir no editor de código.

```
// 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.
```

```
exports.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;
}
```

7. Escolha Save (Salvar).
8. No console do API Gateway, crie uma [API simples \(p. 45\)](#) se você ainda não tiver uma.
9. Selecione sua API na lista de APIs.
10. Selecione Authorizers (Autorizadores).
11. Selecione Create New Authorizer (Criar novo autorizador).
12. Insira um nome para o autorizador.
13. Em Type (Tipo), escolha Lambda.
14. Em Lambda Function (Função do Lambda), selecione a região na qual você criou sua função do autorizador do Lambda e selecione o nome da função na lista suspensa.
15. Deixe Lambda Invoke Role (Função de invocação do Lambda) em branco.
16. Em Lambda Event Payload (Carga de evento do Lambda), selecione Token.
17. Em Token Source (Origem do token), insira **tokenHeader**.
18. Selecione Create (Criar) e Grant & Create (Conceder e criar).
19. Selecione Test (Testar).
20. Para o valor tokenHeader, insira **allow**.

21. Selecione Test (Testar).

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:

```
{  
    "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 403 Forbidden e uma política do IAM Deny que é semelhante ao seguinte, e ocorrerá uma falha na solicitação de método:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Action": "execute-api:Invoke",  
            "Effect": "Deny",  
            "Resource": "arn:aws:execute-api:us-east-1:123456789012:ivdtdhp7b5/ESTestInvoke-  
stage/GET/"  
        }  
    ]  
}
```

- 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.

Note

No código de produção, talvez seja necessário autenticar o usuário antes de conceder a autorização. Se esse for o caso, você também poderá adicionar a lógica de autenticação à função do Lambda e chamar um provedor de autenticação conforme indicado na documentação para esse provedor.

Além de retornar uma política do IAM, a função de autorizador do Lambda também deve retornar o identificador principal do chamador. Opcionalmente, ela também pode retornar um objeto `context` com informações adicionais que podem ser repassadas para o back-end de integração. Para obter mais informações, consulte [Saída de um autorizador do Lambda do Amazon API Gateway \(p. 408\)](#).

EXEMPLO: Criar uma função do autorizador do Lambda com base em solicitação

Para criar uma função Lambda de autorizador com base em solicitação, insira o seguinte código Node.js 8.10 no console do Lambda e teste-o no console do API Gateway da seguinte forma:

1. No console do Lambda, escolha Create function (Criar função).
2. Escolha Author from scratch.
3. Insira um nome para a função.
4. Em Runtime (Tempo de execução), selecione Node.js 8.10.
5. Selecione Create function (Criar função).
6. Copie/cole o código a seguir no editor de código.

```
exports.handler = function(event, context, callback) {
    console.log('Received event:', JSON.stringify(event, null, 2));

    // 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.

    // Retrieve request parameters from the Lambda function input:
    var headers = event.headers;
    var queryStringParameters = event.queryStringParameters;
    var pathParameters = event.pathParameters;
    var stageVariables = event.stageVariables;

    // Parse the input for the parameter values
    var tmp = event.methodArn.split(':');
    var apiGatewayArnTmp = tmp[5].split('/');
    var awsAccountId = tmp[4];
    var region = tmp[3];
    var restApiId = apiGatewayArnTmp[0];
    var stage = apiGatewayArnTmp[1];
    var method = apiGatewayArnTmp[2];
    var resource = '/'; // root resource
    if (apiGatewayArnTmp[3]) {
        resource += apiGatewayArnTmp[3];
    }

    // Perform authorization to return the Allow policy for correct parameters and
    // the 'Unauthorized' error, otherwise.
    var authResponse = {};
    var condition = {};
    conditionIpAddress = {};

    if (headers.HeaderAuth1 === "headerValue1"
        && queryStringParameters.QueryString1 === "queryValue1"
        && stageVariables.StageVar1 === "stageValue1") {
        callback(null, generateAllow('me', event.methodArn));
    } else {
        callback("Unauthorized");
    }
}

// Help function to generate an IAM policy
var generatePolicy = function(principalId, effect, resource) {
    // Required output:
    var authResponse = {};
    authResponse.principalId = principalId;
    if (effect && resource) {
        var policyDocument = {};
        policyDocument.Version = '2012-10-17'; // default version
        policyDocument.Statement = [];
        var statementOne = {};
        statementOne.Action = 'execute-api:Invoke'; // default action
        statementOne.Effect = effect;
        authResponse.policyDocument = policyDocument;
    }
    return authResponse;
}
```

```
        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);
}
```

7. Escolha Save (Salvar).
8. No console do API Gateway, crie uma API simples ([p. 45](#)) se você ainda não tiver uma.
9. Selecione sua API na lista de APIs.
10. Selecione Authorizers (Autorizadores).
11. Selecione Create New Authorizer (Criar novo autorizador).
12. Insira um nome para o autorizador.
13. Em Type (Tipo), escolha Lambda.
14. Em Lambda Function (Função do Lambda), selecione a região na qual você criou sua função do autorizador do Lambda e selecione o nome da função na lista suspensa.
15. Deixe Lambda Invoke Role (Função de invocação do Lambda) em branco.
16. Em Lambda Event Payload (Carga de evento do Lambda), selecione Request (Solicitação).
17. Em Identity Sources (Origens de identidade), adicione um Header (Cabeçalho) chamado **HeaderAuth1**, uma Query String (String de consulta) chamada **QueryString1** e uma Stage Variable (Variável de estágio) chamada **StageVar1**.
18. Selecione Create (Criar) e Grant & Create (Conceder e criar).
19. Selecione Test (Testar).
20. Em HeaderAuth1, insira **headerValue1**. Em QueryString1, insira **queryValue1**. Em StageVar1, insira **stageValue1**.
21. Selecione Test (Testar).

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:

```
{
    "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 chamada de método.

Note

No código de produção, talvez seja necessário autenticar o usuário antes de conceder a autorização. Se esse for o caso, você também poderá adicionar a lógica de autenticação à função do Lambda e chamar um provedor de autenticação conforme indicado na documentação para esse provedor.

Além de retornar uma política do IAM, a função de autorizador do Lambda também deve retornar o identificador principal do chamador. Opcionalmente, ela também pode retornar um objeto context com informações adicionais que podem ser repassadas para o back-end de integração. Para obter mais informações, consulte [Saída de um autorizador do Lambda do Amazon API Gateway \(p. 408\)](#).

Configurar um autorizador do Lambda usando o console do API Gateway

Depois de criar a função do Lambda e verificar se ela funciona, use as etapas a seguir para configurar o autorizador do Lambda do API Gateway (anteriormente conhecido como o autorizador personalizado) no console do API Gateway.

Para configurar um autorizador do Lambda usando o console do API Gateway

1. Faça login no console do API Gateway.
2. Crie uma nova API ou selecione uma API existente e escolha Authorizers (Autorizadores) na API.
3. Selecione Create New Authorizer (Criar novo autorizador).
4. Em Create Authorizer (Criar autorizador), digite um nome de autorizador no campo de entrada Name (Nome).
5. Em Type (Tipo), escolha a opção Lambda.
6. Em Lambda Function (Função Lambda), selecione uma região e, em seguida, escolha uma função do Lambda do autorizador que está na sua conta.
7. Deixe Lambda Invoke Role (Função de invocação do Lambda) em branco para permitir que o console do API Gateway defina uma política com base em recursos. A política concede ao API Gateway permissões para invocar a função do Lambda de autorizador. Você também pode optar por digitar o nome de uma função do IAM para permitir que o API Gateway invoque a função do Lambda de autorizador. Para obter um exemplo dessa função, consulte [Criar uma função do IAM assumível \(p. 97\)](#).

Se você optar por deixar que o console do API Gateway defina a política baseada em recursos, a caixa de diálogo Add Permission to Lambda Function (Adicionar permissão à função do Lambda) será exibida. Escolha OK. Após a criação da autorização do Lambda, você pode testá-la com valores de token de autorização apropriados para verificar se ela funciona conforme esperado.

8. Em Lambda Event Payload (Carga útil do evento do Lambda), escolha Token para um autorizador TOKEN ou Request (Solicitação) para um autorizador REQUEST. (Isso é o mesmo que definir a propriedade `type` como TOKEN ou REQUEST.)
9. Dependendo da opção da etapa anterior, siga um destes procedimentos:
 - a. Para as opções de Token, faça o seguinte:

- Digite o nome de um cabeçalho em Token Source (Origem do token). O cliente da API cliente deve incluir um cabeçalho desse nome para enviar o token de autorização ao autorizador do Lambda.
- Se preferir, forneça uma instrução RegEx no campo de entrada Token Validation. O API Gateway realizará a validação inicial do token de entrada em relação a essa expressão e invocará o autorizador assim que a validação for bem-sucedida. Isso ajuda a reduzir as chances de ser cobrado por tokens inválidos.
- Em Authorization Caching (Cache da autorização), marque ou desmarque a opção Enabled (Habilitado), dependendo de você desejar ou não armazenar em cache as políticas de autorização geradas pelo autorizador. 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 da políticas. Quando o armazenamento em cache de políticas está habilitado, o nome de cabeçalho especificado em Token Source (Origem do token) se torna a chave de cache.

Note

O valor de TTL padrão é de 300 segundos. O valor máximo é de 3600 segundos. Não é possível aumentar esse limite.

- Para a opção de Request (Solicitação), faça o seguinte:

- Em Identity Sources (Origens de identidade), digite um nome de parâmetro de solicitação de um tipo de parâmetro escolhido. Os tipos de parâmetros com suporte são Header, Query String, Stage Variable e Context. Para adicionar mais origens de identidade, escolha Add Identity Source (Adicionar origem de identidade).

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 de 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 de 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.

- Em Authorization Caching (Cache da autorização), marque ou desmarque a opção Enabled (Habilitado), dependendo de você desejar ou não armazenar em cache as políticas de autorização geradas pelo autorizador. Quando o armazenamento em cache de políticas está habilitado, você pode optar por modificar o valor padrão (300) do TTL. Configurar TTL= 0 desabilita o armazenamento em cache de políticas.

Quando o armazenamento em cache está desativado, não é necessário especificar uma origem de identidade. O API Gateway não executa qualquer validação antes de invocar a função do Lambda do autorizador.

Note

Para habilitar o armazenamento em cache, o autorizador deve retornar uma política que seja aplicável a todos os métodos em uma API. Para impor uma política específica de método, você pode definir o valor de TTL como zero para desabilitar o armazenamento em cache de políticas para a API.

10. Escolha Create (Criar) para criar o novo autorizador do Lambda para a API escolhida.
11. Depois que o autorizador é criado para a API, você tem a opção de testar a invocação do autorizador antes de configurá-lo em um método.

Para o autorizador **TOKEN**, digite um token válido no campo de texto de entrada Identity token (Token de identidade) e escolha Test (Testar). O token será transmitido para a função do Lambda como o cabeçalho especificado na configuração Identity token source (Origem do token de identidade) do autorizador.

Para o autorizador **REQUEST**, digite os parâmetros de solicitação válidos correspondentes às origens de identidade especificadas. Em seguida, escolha Test (Testar).

Além de usar o console do API Gateway, você pode usar a AWS CLI ou um SDK da AWS para o API Gateway a fim de testar a invocação de um autorizador. Para usar a AWS CLI, consulte [test-invoke-authorizer](#).

Note

O teste de invocação de um método e o teste de invocação de um autorizador são processos independentes.

Para testar a invocação de um método usando o console do API Gateway, consulte [Usar o console para testar um método de API REST \(p. 564\)](#). Para testar a invocação de um método usando a AWS CLI, consulte [test-invoke-method](#).

Para testar a invocação de um método e de um autorizador configurado, implante a API e, em seguida, use cURL ou Postman para chamar o método fornecendo o token ou os parâmetros de solicitação necessários.

O procedimento a seguir mostra como configurar um método de API para usar o autorizador do Lambda.

Para configurar um método de API para usar um autorizador do Lambda

1. Volte para a API. Crie um novo método ou escolha um método existente. Se necessário, crie um novo recurso.
2. Em Method Execution (Execução de método), escolha o link Method Request (Solicitação de método).
3. Em Settings (Configurações), expanda a lista suspensa Authorization (Autorização) para selecionar o autorizador do Lambda recém-criado (por exemplo, myTestApiAuthorizer) e escolha o ícone de marca de seleção para salvar a opção.
4. Opcionalmente, enquanto ainda estiver na página Method Request (Solicitação de método), escolha Add header (Adicionar cabeçalho) se também desejar passar o token de autorização personalizado para o back-end. Em Name (Nome), digite um nome de cabeçalho personalizado que corresponde ao nome de Token Source (Origem do token) especificado ao criar o autorizador do Lambda para a API. Em seguida, escolha o ícone de marca de seleção para salvar as configurações. Esta etapa não se aplica a autorizadores REQUEST.
5. Escolha Deploy API (Implantar API) para implantar a API em um estágio. Observe o valor Invoke URL (Invocar URL). Você precisará dele quando chamar a API. Para um autorizador do REQUEST que usa variáveis de estágio, você também deve definir as variáveis de estágio necessárias e especificar seus valores enquanto estiver no Stage Editor (Editor de estágio).

Configurar um autorizador do Lambda usando a AWS CLI ou um AWS SDK

Você também pode configurar um autorizador do Lambda do API Gateway usando a AWS CLI ou um AWS SDK.

Primeiro, você precisa conceder aos principais do IAM na conta que são proprietários da API a permissão para chamar [authorizer:create](#) ou [create-authorizer](#) para criar a função do Lambda usada no autorizador do Lambda. Para conceder essa permissão, você precisa criar a seguinte política do IAM:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "apigateway:POST"  
            ],  
            "Resource": [  
                "arn:aws:apigateway:region::/restapis/restapi_id/authorizers"  
            ],  
            //Create Authorizer operation is allowed only with the following Lambda  
            function  
            "Condition": {  
                "StringEquals": {  
                    "apigateway:AuthorizerUri": "arn:aws:lambda:region:account-  
id:function:lambda-function-name"  
                }  
            }  
        ]  
    ]  
}
```

Entrada para um autorizador do Lambda do Amazon API Gateway

Para um autorizador do Lambda (anteriormente conhecido como autorizador personalizado) do tipo TOKEN, você deve especificar um cabeçalho personalizado como a Token Source (Origem do token) ao 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 de cliente. 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.

Para a função Lambda de autorizador TOKEN de exemplo mostrada na seção anterior, a string `{caller-supplied-token}` é `allow`, `deny`, `unauthorized` ou qualquer outro valor de string. Um valor de string vazio é o mesmo que `unauthorized`. Veja a seguir um exemplo dessa entrada para obter uma política `Allow` no método GET de uma API (`ymy8tbxw7b`) da conta da AWS (123456789012) em qualquer estágio (*).

```
{  
    "type": "TOKEN",  
    "authorizationToken": "allow",  
    "methodArn": "arn:aws:execute-api:us-west-2:123456789012:ymy8tbxw7b/*/GET/"  
}
```

Para um autorizador do Lambda do tipo REQUEST, o API Gateway passa os parâmetros de solicitação necessários para a função do Lambda de autorizador como parte do objeto `event`. Os parâmetros de

solicitação afetados incluem cabeçalhos, parâmetros de caminho, parâmetros de string de consulta, variáveis de estágio e algumas das variáveis de contexto de solicitação. O chamador 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 repassados 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:s4x3opwd6i/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": "...",  
        "X-Forwarded-Proto": "https",  
        "CloudFront-Is-SmartTV-Viewer": "false",  
        "Host": "...execute-api.us-east-1.amazonaws.com",  
        "Accept-Encoding": "gzip, deflate",  
        "X-Forwarded-Port": "443",  
        "X-Amzn-Trace-Id": "...",  
        "Via": "...cloudfront.net (CloudFront)",  
        "X-Amz-Cf-Id": "...",  
        "X-Forwarded-For": "..., ...",  
        "Postman-Token": "...",  
        "cache-control": "no-cache",  
        "CloudFront-Is-Desktop-Viewer": "true",  
        "Content-Type": "application/x-www-form-urlencoded"  
    },  
    "queryStringParameters": {  
        "QueryString1": "queryValue1"  
    },  
    "pathParameters": {},  
    "stageVariables": {  
        "StageVar1": "stageValue1"  
    },  
    "requestContext": {  
        "path": "/request",  
        "accountId": "123456789012",  
        "resourceId": "05c7jb",  
        "stage": "test",  
        "requestId": "...",  
        "identity": {  
            "apiKey": "...",  
            "sourceIp": "..."  
        },  
        "resourcePath": "/request",  
        "httpMethod": "GET",  
        "apiId": "s4x3opwd6i"  
    }  
}
```

}

O `requestContext` é um mapa de pares de chave/valor e corresponde à variável [\\$context \(p. 306\)](#). Seu resultado é dependente da API. O API Gateway pode adicionar novas chaves ao mapa. Para obter mais informações sobre a entrada da função do Lambda na integração de proxy do Lambda, consulte [Formato de entrada de uma função Lambda para integração de proxy \(p. 234\)](#).

Saída de um autorizador do Lambda do Amazon 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 da propriedade `usageIdentifierKey`.

Veja a seguir um exemplo dessa saída.

```
{  
    "principalId": "yyyyyyyy", // The principal user identification associated with the token  
    sent by the client.  
    "policyDocument": {  
        "Version": "2012-10-17",  
        "Statement": [  
            {  
                "Action": "execute-api:Invoke",  
                "Effect": "Allow|Deny",  
                "Resource": "arn:aws:execute-api:{regionId}:{accountId}:{apiId}/{stage}/{httpVerb}/  
[{resource}][{child-resources}]"  
            }  
        ]  
    },  
    "context": {  
        "stringKey": "value",  
        "numberKey": "1",  
        "booleanKey": "true"  
    },  
    "usageIdentifierKey": "{api-key}"  
}
```

Aqui, uma instrução da 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`). 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 IAM para executar a API no API Gateway \(p. 386\)](#).

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 back-

end. Para obter mais informações, consulte [Variáveis \\$context para modelos de dados, autorizadores, modelos de mapeamento e registro de acesso em logs do CloudWatch \(p. 306\)](#).

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 back-end usando um modelo de mapeamento de solicitação de integração. Isso permite que o back-end 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 do Lambda de back-end 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 “Utilização de planos de uso com chaves de API” \(p. 450\)](#).

Veja a seguir a saída de exemplo do autorizador do Lambda de exemplo. A saída de exemplo contém uma instrução da política para bloquear chamadas (Deny) ao método GET em uma API (`yml8tbxw7b`) de uma conta da AWS (123456789012) em qualquer estágio (*).

```
{
  "principalId": "user",
  "policyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": "execute-api:Invoke",
        "Effect": "Deny",
        "Resource": "arn:aws:execute-api:us-west-2:123456789012:yml8tbxw7b/*/*/*"
      }
    ]
  }
}
```

Chamar uma API com autorizadores do Lambda do API Gateway

Depois de configurar o autorizador do Lambda (anteriormente conhecido como autorizador personalizado) e implantar a API, teste 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 forem definidas ou forem nulas ou vazias.

No próximo exemplo, mostramos como usar Postman para chamar ou testar a API com o autorizador `TOKEN` do Lambda habilitado descrito anteriormente. 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**. Escolha Send.

The screenshot shows the Postman interface. At the top, the URL is set to `https://<api-id>.execute-api.<region>.amazonaws.com/test`. In the 'Authorization' section of the Headers tab, there is a key 'Auth' with the value 'allow'. The response status is 200 OK. The response body is a JSON object:

```
1 [ {  
2   "args": {},  
3   "headers": {  
4     "Accept": "application/json",  
5     "Host": "httpbin.org",  
6     "User-Agent": "AmazonAPIGateway_y_<api-id>",  
7     "X-Amzn-Apigateway-Api-Id": "y_<api-id>"  
8   },  
9   "origin": "54.186.57.107",  
10  "url": "http://httpbin.org/get"  
11 } ]
```

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**. Escolha Send.

The screenshot shows the Postman interface with a GET request to `https://<api-id>.execute-api.<region>.amazonaws.com/test`. The Authorization tab is selected, showing a Header named "Auth" with the value "deny". The response status is 403 Forbidden, and the body contains the message: `"Message": "User is not authorized to access this resource"`.

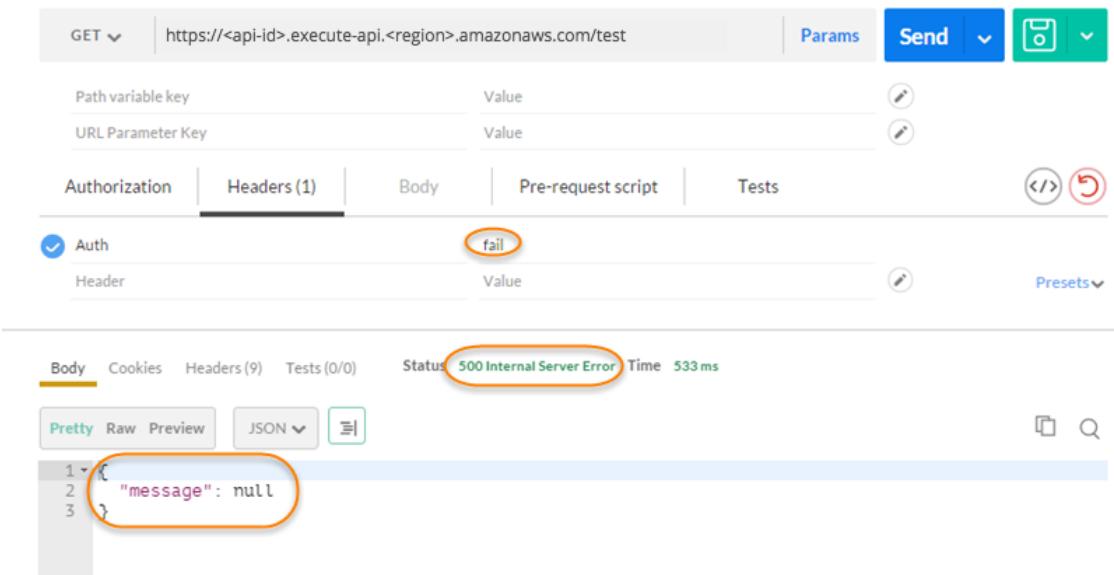
A resposta mostra que o autorizador do Lambda do API Gateway retorna uma resposta 403 Forbidden sem autorizar a chamada para acessar o endpoint HTTP.

3. No Postman, altere o valor do cabeçalho do token de autorização do Lambda para `unauthorized` e escolha Send (Enviar).

The screenshot shows the Postman interface with a GET request to `https://<api-id>.execute-api.<region>.amazonaws.com/test`. The Authorization tab is selected, showing a Header named "Auth" with the value "unauthorized". The response status is 401 Unauthorized, and the body contains the message: `"message": "Unauthorized"`.

A resposta mostra que o API Gateway retorna uma resposta 401 Unauthorized (401 Não autorizado) sem autorizar a chamada para acessar o endpoint HTTP.

4. Agora, altere o valor do cabeçalho do token de autorização do Lambda para `fail`. Escolha Send.



A resposta mostra que o API Gateway retorna uma resposta 500 Internal Server Error sem autorizar a chamada a acessar o endpoint HTTP.

Configurar um autorizador do Lambda 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 em que o Amazon API Gateway estiver 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 a função do Lambda de autorizador entre contas usando o console do Amazon API Gateway.

Estas instruções pressupõem que você já tenha uma API do API Gateway em uma conta da AWS e uma função do Lambda de autorizador em outra conta.

Configurar um autorizador do Lambda entre contas usando o console do API Gateway

Faça o login no console do Amazon API Gateway na sua primeira conta (aquele que contém a sua API) e faça o seguinte:

1. Localize sua API e escolha Authorizers (Autorizadores).
2. Selecione Create New Authorizer (Criar novo autorizador).
3. Em Create Authorizer (Criar autorizador), digite um nome de autorizador no campo de entrada Name (Nome).
4. Em Type (Tipo), escolha a opção Lambda.
5. Para Lambda Function (Função do Lambda), copie e cole o ARN completo para a função do Lambda do autorizador na sua segunda conta.

Note

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

6. Deixe Lambda Invoke Role (Função de invocação do Lambda) em branco para permitir que o console do API Gateway defina uma política com base em recursos. A política concede ao API Gateway permissões para invocar a função do Lambda de autorizador. Você também pode optar por digitar o nome de uma função do IAM para permitir que o API Gateway invoque a função do Lambda de autorizador. Para obter um exemplo dessa função, consulte [Criar uma função do IAM assumível \(p. 97\)](#).

Se você optar por deixar que o console do API Gateway defina a política baseada em recursos, a caixa de diálogo Add Permission to Lambda Function (Adicionar permissão à função do Lambda) será exibida. Escolha OK. Após a criação da autorização do Lambda, você pode testá-la com valores de token de autorização apropriados para verificar se ela funciona conforme esperado.

7. Em Lambda Event Payload (Carga útil do evento do Lambda), escolha Token para um autorizador TOKEN ou Request (Solicitação) para um autorizador REQUEST.
8. Dependendo da opção escolhida na etapa anterior, siga um destes procedimentos:

- a. Para as opções de Token, faça o seguinte:
 - i. Digite o nome de um cabeçalho em Token Source (Origem do token). O cliente da API cliente deve incluir um cabeçalho desse nome para enviar o token de autorização ao autorizador do Lambda.
 - ii. Se preferir, forneça uma instrução RegEx no campo de entrada Token Validation (Validação do token). O API Gateway realizará a validação inicial do token de entrada em relação a essa expressão e invocará o autorizador assim que a validação for bem-sucedida. Isso ajuda a reduzir as chances de ser cobrado por tokens inválidos.
 - iii. Em Authorization Caching (Cache da autorização), marque ou desmarque a opção Enabled (Habilitado), dependendo de você desejar ou não armazenar em cache as políticas de autorização geradas pelo autorizador. Quando o armazenamento em cache de políticas está habilitado, você pode optar por modificar o valor padrão (300) do TTL. Configurar TTL= 0 desabilita o armazenamento em cache de políticas. Quando o armazenamento em cache de políticas está habilitado, o nome de cabeçalho especificado em Token Source (Origem do token) se torna a chave de cache.
- b. Para a opção de Request (Solicitação), faça o seguinte:
 - i. Em Identity Sources (Origens de identidade), digite um nome de parâmetro de solicitação de um tipo de parâmetro escolhido. Os tipos de parâmetros com suporte são Header, Query String, Stage Variable e Context. Para adicionar mais origens de identidade, escolha Add Identity Source (Adicionar origem de identidade).

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 de 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 de 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.

- ii. Em Authorization Caching (Cache da autorização), deixe a opção Enabled (Habilitado) selecionada. Deixe o valor TTL definido como padrão (300).

9. Escolha Create (Criar) para criar o novo autorizador do Lambda para a API escolhida.
10. Você verá uma janela pop-up que diz Adicionar permissão à função do Lambda: você selecionou uma função do Lambda de outra conta. Verifique se você tem uma política de função adequada para essa função. Você pode fazer isso executando o seguinte comando da CLI da AWS da conta **123456789012**, seguido por uma string de comando aws lambda add-permission.

11. Copie e cole a string de comando `aws lambda add-permission` em uma janela da AWS CLI configurada para sua segunda conta. Isso concederá acesso para sua primeira conta à função Lambda do autorizador da sua segunda conta.
12. Na janela pop-up da etapa anterior, escolha OK.

Controle o acesso à API REST usando o Grupos de usuários do Amazon Cognito como um autorizador

Em vez de usar [funções e políticas do IAM \(p. 378\)](#) ou [autorizadores do Lambda \(p. 396\)](#) (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, em seguida, configurar um método da API para usar esse autorizador. Após a implantação da API, o cliente deve primeiro assinar o usuário no grupo de usuários, obter uma [identidade ou token de acesso](#) para o usuário e, em seguida, chamar o método de API com um dos tokens, que são normalmente definidos para o cabeçalho da solicitação `Authorization`. 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 do — 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 um 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 token de acesso.
- Use uma estrutura específica 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

- [Obter permissões para criar autorizadores de grupo de usuários do Amazon Cognito para uma API REST \(p. 415\)](#)
- [Criar um grupo de usuários do Amazon Cognito para uma API REST \(p. 416\)](#)
- [Integrar uma API REST a um grupo de usuários do Amazon Cognito \(p. 417\)](#)
- [Chamar uma API REST integrada a um grupo de usuários do Amazon Cognito \(p. 421\)](#)
- [Configurar o autorizador do Amazon Cognito entre contas para uma API REST usando o console do API Gateway \(p. 422\)](#)

Obter permissões para criar autorizadores de grupo de usuários do Amazon Cognito para uma API REST

Para criar um autorizador com um grupo de usuários do Amazon Cognito, você deve ter permissões Allow para criar ou atualizar um autorizador com o grupo de usuários do Amazon Cognito escolhido. A seguir, um documento de política do IAM mostra um exemplo de tais permissões:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "apigateway:POST"  
            ],  
            "Resource": "arn:aws:apigateway:*:::/restapis/*/authorizers",  
            "Condition": {  
                "ArnLike": {  
                    "apigateway:CognitoUserPoolProviderArn": [  
                        "arn:aws:cognito-idp:us-east-1:123456789012:userpool/us-  
east-1_aD06NQmjO",  
                        "arn:aws:cognito-idp:us-east-1:234567890123:userpool/us-  
east-1_xJ1MQtPEN"  
                    ]  
                }  
            }  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "apigateway:PATCH"  
            ],  
            "Resource": "arn:aws:apigateway:*:::/restapis/*/authorizers/*",  
            "Condition": {  
                "ArnLike": {  
                    "apigateway:CognitoUserPoolProviderArn": [  
                        "arn:aws:cognito-idp:us-east-1:123456789012:userpool/us-  
east-1_aD06NQmjO",  
                        "arn:aws:cognito-idp:us-east-1:234567890123:userpool/us-  
east-1_xJ1MQtPEN"  
                    ]  
                }  
            }  
        }  
    ]  
}
```

Certifique-se de que a política está anexada ao seu usuário do IAM, um grupo do IAM ao qual você pertence ou a uma função do IAM, à qual você está atribuído.

No documento de política anterior, a ação `apigateway:POST` destina-se à criação de um novo autorizador e a ação `apigateway:PATCH` destina-se à atualização de um autorizador existente. Você pode restringir a política a uma região específica ou a uma determinada API ao sobreescriver os dois primeiros caracteres curinga (*) dos valores `Resource`, respectivamente.

As cláusulas `Condition` que são usadas aqui são para restringir as permissões `Allowed` para os grupos de usuários especificados. Quando uma cláusula `Condition` está presente, o acesso a qualquer grupo de usuários que não corresponda às condições é negado. Quando uma permissão não tem uma cláusula `Condition`, o acesso a qualquer grupo de usuários é permitido.

Você tem as seguintes opções para definir a cláusula `Condition`:

- Você pode definir uma expressão condicional `ArnLike` ou `ArnEquals` para permitir a criação ou atualização de autorizadores `COGNITO_USER_POOLS` somente com os grupos de usuários especificados.
- Você pode definir uma expressão condicional `ArnNotLike` ou `ArnNotEquals` para permitir a criação ou atualização de autorizadores `COGNITO_USER_POOLS` com qualquer grupo de usuários não especificado na expressão.
- Você pode omitir a cláusula `Condition` para permitir a criação ou atualização de autorizadores `COGNITO_USER_POOLS` com qualquer grupo de usuários, de uma conta da AWS e em qualquer região.

Para obter mais informações sobre as expressões condicionais do nome de recurso da Amazon (ARN), consulte [Operadores de condição do nome de recurso](#) da Amazon. Como mostrado no exemplo, `apigateway:CognitoUserPoolProviderArn` é uma lista de ARNs do grupo de usuários `COGNITO_USER_POOLS` que pode ou não ser usado com um autorizador do API Gateway do tipo `COGNITO_USER_POOLS`.

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. Para obter instruções sobre como criar um grupo de usuários, consulte [Configuração de grupos de usuários](#) no Guia do desenvolvedor do Amazon Cognito.

Note

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`.

PetStoreUsers

General settings
Users and groups
Attributes
Policies
MFA and verifications
Advanced security beta
Message customizations
Tags
Devices
App clients
Triggers
Analytics
App integration
App client settings
Domain name
UI customization
Resource servers
Federation
Identity providers
Attribute mapping

Do you want to define resource servers and custom scopes?

You can define resource servers and custom scopes for OAuth 2.0 flows. [Learn more about resource servers and scopes.](#)

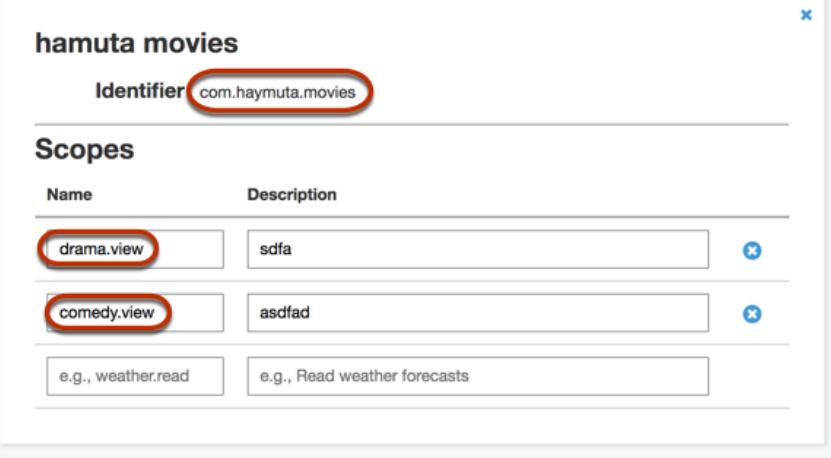
hamuta movies

Identifier com.haymuta.movies

Scopes

Name	Description
drama.view	sdfa
comedy.view	asdfad
e.g., weather.read	e.g., Read weather forecasts

Add another resource server Configure app client settings



Integrar uma API REST a um grupo de usuários do Amazon Cognito

Após criar um grupo de usuários do Amazon Cognito no API Gateway, você deverá criar um autorizador do COGNITO_USER_POOLS que use o grupo de usuários. O procedimento a seguir percorre as etapas necessárias para fazer isso usando o console do API Gateway.

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 [Implantando uma API REST no Amazon API Gateway \(p. 516\)](#).

Para 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, escolha Authorizers (Autorizadores) na API especificada.
3. Em Authorizers (Autorizadores), escolha Create New Authorizer (Criar novo autorizador).
4. Para configurar o novo autorizador para usar um grupo de usuários, faça o seguinte:
 - a. Digite um nome de autorizador em Name (Nome).
 - b. Selecione a opção Cognito.
 - c. Escolha uma região em Cognito User Pool (Grupo de usuários do Cognito).
 - d. Selecione um grupo de usuários disponível. Você deve ter criado um grupo de usuários para a região selecionada no Amazon Cognito para que ele seja exibido na lista suspensa.
 - e. Em Token source (Origem do token), digite 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.

- f. Opcionalmente, digite uma expressão regular no campo Token validation (Validação do token) para validar o campo aud (público) do token de identidade antes de a solicitação ser autorizada com o Amazon Cognito.
 - g. Para concluir a integração do grupo de usuários com a API, escolha Create (Criar).
5. Após criar o autorizador COGNITO_USER_POOLS, você tem a opção de testar a invocação fornecendo um token de identidade que é provisionado do grupo de usuários. Você pode obter esse token de identidade chamando o [SDK de identidade do Amazon Cognito](#) para fazer login do usuário. Certifique-se de usar o token de identidade retornado, não o token de acesso.

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. O próximo procedimento mostra as etapas para configurar o autorizador em um método de API.

Para configurar um autorizador do **COGNITO_USER_POOLS** nos métodos

1. Selecione (ou crie) um método na sua API.
2. Escolha Method Request (Solicitação de método).
3. Em Settings (Configurações), escolha o ícone de lápis ao lado de Authorization (Autorização).
4. Escolha um dos Amazon Cognito user pool authorizers (Autorizadores do grupo de usuários do Amazon Cognito) disponíveis na lista suspensa.
5. Para salvar as configurações, escolha o ícone de marca de seleção.
6. Para usar um token de identidade, faça o seguinte:
 - a. Deixe a opção OAuth Scopes (Escopos OAuth) não especificada (como NONE).
 - b. Se necessário, selecione Integration Request (Solicitação de integração) para adicionar 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 back-end:

```
{  
  "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']"  
  }  
}
```

7. Para usar um token de acesso, faça o seguinte:

- a. Selecione o ícone de lápis ao lado de OAuth Scopes (Escopos OAuth).
- b. Digite 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 \(p. 416\)](#), um dos escopos é com.hamuta.movies/drama.view. Use um único espaço para separar vários escopos.

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.

- c. Para salvar a configuração, escolha o ícone de marca de seleção.
8. Repita essas etapas para outros métodos que você quiser.

Com o autorizador COGNITO_USER_POOLS, se a opção OAuth Scopes (Escopos OAuth) 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": ["access_token"]  
                    }  
                ],  
                "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"  
                    }  
                },  
                ...  
            }  
        }  
    }  
}
```

```
"get": {
    "consumes": [
        "application/json"
    ],
    "produces": [
        "text/html"
    ],
    "responses": {
        "200": {
            "description": "200 response",
            "headers": {
                "Content-Type": {
                    "type": "string"
                }
            }
        }
    },
    "security": [
        {
            "MyUserPool": ["com.hamuta.movies/drama.view", "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 do API Gateway para OpenAPI \(p. 606\)](#).

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.
- Obter um token de identidade do usuário conectado no grupo de usuários.
- Incluir o token de identidade no cabeçalho `Authorization` (ou outro cabeçalho especificado quando você criou o autorizador).

Você pode usar um dos [SDKs da AWS](#) para realizar essas tarefas. Por exemplo:

- Para usar o SDK do Android, consulte [Configurando o AWS Mobile SDK for Android para trabalhar com grupos de usuários](#).
- Para usar o SDK do iOS, consulte [Configurando o AWS Mobile SDK for iOS para trabalhar com grupos de usuários](#).
- Para usar o JavaScript, consulte [Configurando o AWS SDK for JavaScript no navegador para trabalhar com grupos de usuários](#).

O procedimento a seguir descreve as etapas para realizar essas tarefas. Para obter mais informações, consulte as postagens de blog em [Usando o Android SDK com grupos de usuários do Amazon Cognito e Usando o grupo de usuários do Amazon Cognito para o iOS](#).

Para chamar uma API que é integrada a um grupo de usuários

1. Inscreva um usuário inicial em um grupo de usuários especificado.
2. Conecte um usuário ao grupo de usuários.
3. Obtenha o token de identidade do usuário.
4. Chame métodos de API que são configurados com um autorizador de grupo de usuários e forneça o token não expirado no cabeçalho `Authorization` ou em outro cabeçalho de sua escolha.
5. Quando o token expirar, repita as etapas de 2 – 4. Toques de identidade provisionados pelo Amazon Cognito expiram em uma hora.

Para obter exemplos de códigos, consulte uma [amostra de Java para Android](#) e uma [amostra de Objective-C para iOS](#).

Configurar o autorizador do Amazon Cognito entre contas para uma API REST usando o console do API Gateway

Agora também é possível usar um grupo de usuários do Amazon Cognito de uma conta diferente da AWS como autorizador da API. Cada conta pode estar em qualquer região em que o Amazon API Gateway estiver disponível. O grupo de usuários do Amazon Cognito pode usar estratégias de autenticação de token de portador, como OAuth ou SAML. Isso facilita o gerenciamento centralizado e o compartilhamento de um autorizador de grupo de usuários do Amazon Cognito central 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.

Estas instruções pressupõem que você já tenha 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 usando o console do API Gateway

Faça o login no console do Amazon API Gateway na sua primeira conta (aquele que contém a sua API) e faça o seguinte:

1. Localize sua API e escolha Authorizers (Autorizadores).
2. Selecione Create New Authorizer (Criar novo autorizador).
3. Em Create Authorizer (Criar autorizador), digite um nome de autorizador no campo de entrada Name (Nome).
4. Em Type (Tipo), selecione a opção Cognito.
5. Em Cognito User Pool (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).

6. Digite o nome de um cabeçalho em Token Source (Origem do token). O cliente da API cliente deve incluir um cabeçalho desse nome para enviar o token de autorização ao autorizador do Amazon Cognito.
7. Se preferir, forneça uma instrução RegEx no campo de entrada Token Validation (Validação do token). O API Gateway realizará a validação inicial do token de entrada em relação a essa expressão e invocará o autorizador assim que a validação for bem-sucedida. Isso ajuda a reduzir as chances de ser cobrado por tokens inválidos.
8. Selecione Create (Criar) para criar o novo autorizador do Amazon Cognito para a API.

Habilitar o CORS para um recurso da API REST do 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. Se os recursos da API REST receberem solicitações HTTP não simples de origem cruzada, será necessário ativar o suporte ao CORS.

Tópicos

- [Determinar se deseja habilitar o suporte ao CORS \(p. 423\)](#)
- [O que significa habilitar o suporte ao CORS \(p. 424\)](#)
- [Testar o CORS \(p. 426\)](#)
- [Habilitar o CORS em um recurso usando o console do API Gateway \(p. 426\)](#)
- [Habilitar o CORS em um recurso usando a API de importação do API Gateway \(p. 428\)](#)

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`)

As solicitações HTTP entre origens podem ser divididas em dois tipos: solicitações simples e solicitações não 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 POST, GET e HEAD.
- 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 de método POST simples entre origens, a resposta do seu recurso precisa incluir o cabeçalho Access-Control-Allow-Origin, onde o valor da chave de cabeçalho é definido como '*' (qualquer origem) ou é definido como as origens com permissão para acessar esse recurso.

Todas as outras solicitações HTTP entre origens são solicitações não simples. Se os recursos da sua API recebem solicitações não simples, será necessário habilitar o suporte ao CORS.

O que significa habilitar o suporte ao CORS

Quando um navegador recebe uma solicitação HTTP simples, o [protocolo CORS](#) exige que o navegador envie uma solicitação de simulação para o servidor e aguarde a aprovação (ou uma solicitação de credenciais) do servidor antes de enviar a solicitação real. A solicitação de simulação é exibida para sua API como uma solicitação HTTP que:

- Inclui um cabeçalho Origin.
- Usa o método OPTIONS.
- Inclui os seguintes cabeçalhos:
 - Access-Control-Request-Method
 - Access-Control-Request-Headers

Para oferecer suporte ao CORS, portanto, um recurso de API REST precisa implementar um método OPTIONS que possa responder à solicitação de simulação OPTIONS com pelo menos os seguintes cabeçalhos de resposta exigidos pela Busca padrão:

- Access-Control-Allow-Methods
- Access-Control-Allow-Headers
- Access-Control-Allow-Origin

A forma como você habilita o suporte ao CORS depende do tipo de integração da sua API.

Ativar o suporte ao CORS para integrações simuladas

Para uma integração simulada, ative o CORS criando um método OPTIONS para retornar os cabeçalhos de resposta necessários (com valores estáticos adequados) como cabeçalhos de resposta de método. Além disso, cada um dos métodos reais ativados para CORS também devem retornar o cabeçalho Access-Control-Allow-Origin: '*request-originating server addresses*' pelo menos em suas respostas 200, onde o valor da chave de cabeçalho é definido como '*' (qualquer origem) ou é definido como as origens com permissão para acessar o recurso.

Ativar suporte ao CORS para integrações não-proxy ou HTTP do Lambda e integrações de serviços da AWS

Para obter uma integração personalizada (não proxy) do Lambda, integração personalizada (não proxy) HTTP ou integração de serviços da AWS, você pode configurar os cabeçalhos necessários usando a resposta de método do API Gateway e as configurações de resposta de integração. O API Gateway criará um método OPTIONS e tentará adicionar o cabeçalho Access-Control-Allow-Origin às respostas de integração de método existentes. Isso não funciona sempre, e, às vezes, é necessário modificar manualmente a resposta de integração para habilitar o CORS corretamente. Normalmente, isso apenas significa modificar manualmente a resposta de integração para retornar o cabeçalho Access-Control-Allow-Origin.

Ativar suporte ao CORS para integrações de proxy HTTP ou do Lambda

Para uma integração de proxy do Lambda ou integração de proxy HTTP, você ainda pode configurar os cabeçalhos de resposta OPTIONS necessários no API Gateway. No entanto, seu back-end é responsável

por retornar os cabeçalhos `Access-Control-Allow-Headers` e `Access-Control-Allow-Origin`, pois uma integração de proxy não retorna uma resposta de integração.

Veja a seguir um exemplo de uma função Lambda Node.js que é configurada para retornar os cabeçalhos CORS necessários:

```
'use strict';

exports.handler = function(event, context) {

    var responseCode = 200;

    var response = {
        statusCode: responseCode,
        headers: {
            "x-custom-header" : "my custom header value",
            "Access-Control-Allow-Origin": "my-origin.com"
        },
        body: JSON.stringify(event)
    };

    context.succeed(response);
};
```

Um exemplo de Node.js mais completa pode ser encontrada em https://github.com/awslabs/serverless-application-model/blob/master/examples/2016-10-31/api_swagger_cors/index.js.

Veja a seguir um exemplo de um trecho de código Python que retorna os cabeçalhos CORS necessários:

```
response[ "headers" ] = {
    'Content-Type': 'application/json',
    'Access-Control-Allow-Origin': '*'
}
```

Veja a seguir um exemplo que retorna os cabeçalhos necessários para o CORS usando o Serverless Application Model (SAM), incluindo `AllowHeaders`:

```
Globals:
  Api:
    # Allows an application running locally on port 8080 to call this API
    Cors:
      AllowMethods: "'OPTIONS,POST,GET'"
      AllowHeaders: "'Content-Type'"
      AllowOrigin: "'http://localhost:8080'"
```

Veja a seguir um exemplo de proxy do Lambda que retorna os mesmos cabeçalhos que o exemplo do SAM:

```
return {
    'statusCode': 200,
    'headers': {
        "Access-Control-Allow-Origin": "http://localhost:8080",
        "Access-Control-Allow-Headers": "Content-Type",
        "Access-Control-Allow-Methods": "OPTIONS,POST,GET"
    },
    'body': json.dumps(response)
}
```

Testar o CORS

Você pode testar o CORS personalizando o comando `cURL` a seguir para o método real e o cabeçalho de origem que você está usando:

```
curl -v -X OPTIONS -H "Access-Control-Request-Method: POST" -H "Origin: http://example.com"  
https://{restapi_id}.execute-api.{region}.amazonaws.com/{stage_name}
```

Habilitar 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.

Important

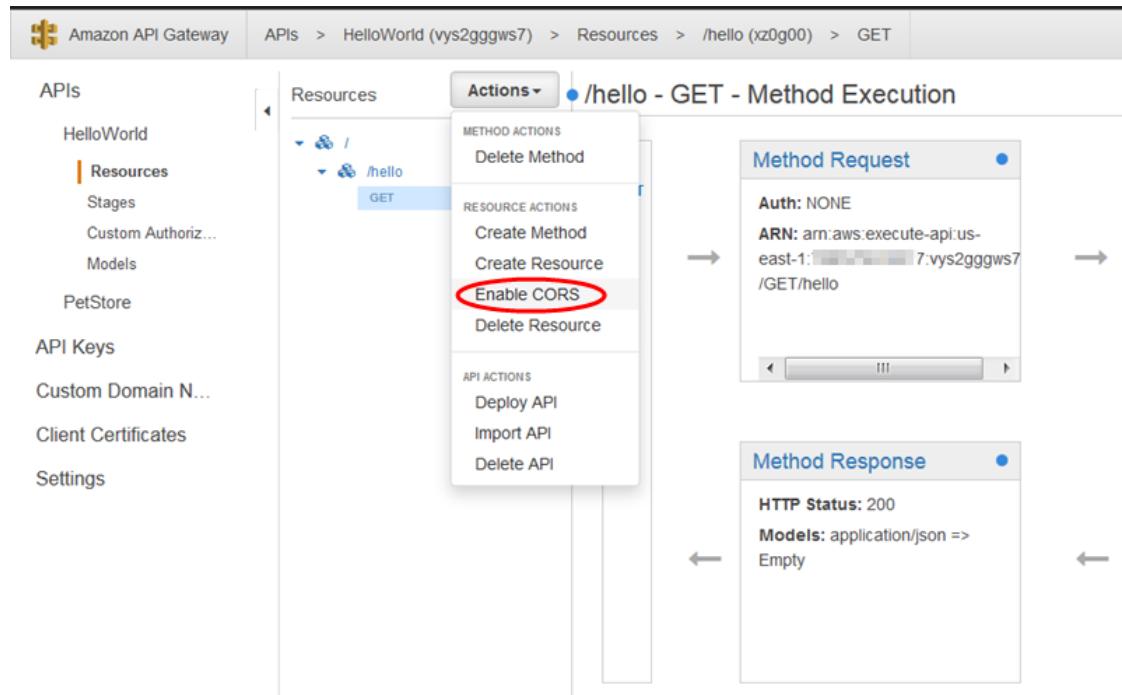
Os recursos podem conter recursos filhos. A habilitação do suporte ao CORS para um recurso e seus métodos não habilita recursivamente para recursos filhos e seus métodos.

Habilitar o suporte ao CORS em um recurso de API REST

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione a API na lista de APIs.
3. Escolha um recurso em Recursos. Isso habilitará o CORS para todos os métodos no recurso.

Como alternativa, você pode escolher um método no recurso para habilitar o CORS apenas para esse método.

4. Escolha Habilitar CORS no menu suspenso Ações.



5. No formulário Habilitar CORS, faça o seguinte:
 - a. No campo de entrada Access-Control-Allow-Headers, digite 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 as origens a serem permitidas para acessar o recurso.
 - Escolha Ativar CORS e substituir cabeçalhos CORS existentes.

The screenshot shows the 'Enable CORS' configuration for a resource named '/hello'. The 'Actions' dropdown is open, and the 'Enable CORS' option is selected. The main panel describes CORS and its purpose. Under the 'Methods' section, 'GET' and 'OPTIONS' are selected. The 'Access-Control-Allow-Methods' field contains 'GET,OPTIONS'. The 'Access-Control-Allow-Headers' field contains "'Content-Type,X-Amz-Date,Authorization'". The 'Access-Control-Allow-Origin' field is empty and has a warning icon. An 'Advanced' section is collapsed. A large blue button at the bottom right says 'Enable CORS and replace existing CORS headers'.

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 back-end deve retornar os cabeçalhos CORS aplicáveis, como Access-Control-Allow-Origin.

- Em Confirmar alterações de método, escolha Sim, substituir valores existentes para confirmar a novas configurações de CORS.

The screenshot shows a confirmation dialog titled 'Confirm method overwrite'. It lists several changes that will be made to the resource's methods, including creating an OPTIONS method, adding 200 Method Response with Empty Response Model to the OPTIONS method, and adding Mock Integration to the OPTIONS method. It also lists changes to the GET method, such as adding 200 Integration Response, adding Access-Control-Allow-Headers, Access-Control-Allow-Methods, Access-Control-Allow-Origin, and adding Header Mappings to the OPTIONS method. A note states that the 'Access-Control-Allow-Origin' header will not be removed if it already exists. At the bottom, there are 'Cancel' and 'Yes, overwrite existing values' buttons, and an 'Enable' button is visible in the background.

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 “Implantar uma API REST no console” \(p. 518\)](#).

Habilitar o CORS em um recurso usando a API de importação do API Gateway

Se você estiver usando o recurso [Import API do API Gateway \(p. 356\)](#), 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.

O exemplo a seguir cria um método OPTIONS para uma integração simulada.

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:
        application/json: |
          {
            "statusCode" : 200
          }
      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 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 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 : "'*'"
```

Gerar e configurar um certificado SSL para autenticação de back-end

Você pode usar o API Gateway para gerar um certificado SSL e usar sua chave pública no back-end para verificar se as solicitações HTTP para seu sistema back-end são provenientes do API Gateway. Isso permite que o back-end HTTP controle e aceite apenas solicitações provenientes do Amazon API Gateway, mesmo que o back-end seja acessível publicamente.

Note

Alguns servidores de back-end talvez não ofereçam suporte à autenticação de cliente SSL como o API Gateway, e podem retornar um erro de certificado SSL. Para obter uma lista de servidores de back-end incompatíveis, consulte [the section called “Observações importantes” \(p. 730\)](#).

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 através das APIs.

Tópicos

- [Gerar um certificado de cliente usando o console do API Gateway \(p. 430\)](#)
- [Configurar uma API para usar certificados SSL \(p. 430\)](#)
- [Testar chamada para verificar a Configuração de certificado do cliente \(p. 430\)](#)
- [Configurar servidor HTTPS de back-end para verificar o certificado do cliente \(p. 431\)](#)
- [Girar um certificado de cliente prestes a expirar \(p. 431\)](#)
- [Autoridades de certificado com suporte pelo API Gateway para integrações HTTP e de proxy HTTP \(p. 432\)](#)

Gerar um certificado de cliente usando o console do API Gateway

1. No painel de navegação principal, escolha Client Certificates (Certificados do cliente).
2. No painel Client Certificates (Certificados do cliente), escolha Generate Client Certificate (Gerar certificado do cliente).
3. Opcionalmente, para Edit (Editar), adicione um título descritivo para o certificado gerado e escolha Save (Salvar) para salvar a descrição. 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 supõem que você já concluiu [Gerar um certificado de cliente usando o console do API Gateway \(p. 430\)](#).

1. No console do API Gateway, crie ou abra uma API para a qual você deseja usar o certificado de cliente. Certifique-se que a API tenha sido implantada em um estágio.
2. Escolha Stages (Estágios) na API selecionada e selecione um estágio.
3. No Stage Editor (Editor de estágios), selecione um certificado na seção Client Certificate (Certificado do cliente).
4. Para salvar as configurações, escolha Save Changes (Salvar alterações).

Se a API tiver sido implantada anteriormente no console do API Gateway, será necessário reimplantá-la para que as alterações entrem em vigor. Para obter mais informações, consulte [the section called "Reimplantar uma API REST em um estágio" \(p. 519\)](#).

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. Em Client (Cliente), escolha Test (Testar).
2. Em Client Certificate (Certificado do cliente), selecione Test (Testar) para invocar a solicitação de método.

O API Gateway apresenta o certificado SSL escolhido para que o back-end HTTP autentique a API.

Configurar servidor HTTPS de back-end para verificar o certificado do cliente

Estas instruções supõem que você já tenha concluído [Gerar um certificado de cliente usando o console do API Gateway \(p. 430\)](#) e baixado uma cópia do certificado do cliente. Para fazer download de um certificado de cliente, chame `clientcertificate:by-id` da API REST do API Gateway ou `get-client-certificate` da AWS CLI.

Antes de configurar um servidor HTTPS de back-end para verificar o certificado SSL do cliente do API Gateway, você deve 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" \(p. 432\)](#).

Como exemplo, suponha que o arquivo de certificado do cliente seja `apig-cert.pem` e que a chave privada do servidor e os arquivos de certificado sejam `server-key.pem` e `server-cert.pem`, respectivamente, para um servidor Node.js no back-end, você pode configurar o servidor da seguinte forma:

```
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. Para verificar a data de expiração do certificado, chame `clientCertificate:by-id` da API REST do API Gateway ou o comando `get-client-certificate` da AWS CLI e inspecione a propriedade `expirationDate` retornada.

Para girar um certificado de cliente, siga as etapas abaixo:

1. Para gerar um novo certificado de cliente, chame `clientcertificate:generate` da API REST do API Gateway ou o comando `generate-client-certificate` da AWS CLI. Neste tutorial, vamos supor que o novo ID de certificado do cliente é `ndiqef`.
2. Atualize o servidor de back-end para incluir o novo certificado do cliente. Ainda não remova o certificado de cliente existente.

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. Atualize o estágio da API para usar o novo certificado do cliente chamando [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"
    }
  ]
}
```

ou chame o comando da CLI [update-stage](#).

4. Atualize o servidor de back-end para remover o certificado antigo.
5. Exclua o certificado antigo do API Gateway chamando [clientcertificate:delete](#) da API REST do API Gateway, especificando o clientCertificateId (a1b2c3) do certificado antigo:

```
DELETE /clientcertificates/a1b2c3
```

ou chamando o comando da CLI [delete-client-certificate](#):

```
aws apigateway delete-client-certificate --client-certificate-id a1b2c3
```

Para mudar um certificado do cliente no console para uma API implantada anteriormente:

1. No painel de navegação principal, escolha Client Certificates (Certificados do cliente).
2. No painel Client Certificates (Certificados do cliente), escolha Generate Client Certificate (Gerar certificado do cliente).
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. No painel Stage Editor (Editor de estágios), selecione o novo certificado na seção Client Certificate (Certificado do cliente).
6. Para salvar as configurações, escolha Save Changes (Salvar alterações).

Você precisará implantar a API novamente para que as alterações entrem em vigor. Para obter mais informações, consulte [the section called “Reimplantar uma API REST em um estágio” \(p. 519\)](#).

Autoridades de certificado com suporte pelo API Gateway para integrações HTTP e de proxy HTTP

A lista a seguir mostra as autoridades de certificação com suporte pelo API Gateway para integrações HTTP e de proxy HTTP.

```
Alias name: mozillacert81.pem
MD5: D5:E9:81:40:C5:18:69:FC:46:2C:89:75:62:0F:AA:78
```

```
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:C4:40:8E  
Alias name: mozillacert99.pem  
MD5: 2B:70:20:56:86:82:A0:18:C8:07:53:12:28:70:21:72  
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:50  
Alias name: swisssignplatinumg2ca  
MD5: C9:98:27:77:28:1E:3D:0E:15:3C:84:00:B8:85:03:E6  
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:14:36  
Alias name: mozillacert145.pem  
MD5: 60:84:7C:5A:CE:DB:0C:D4:CB:A7:E9:FE:02:C6:A9:C0  
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:A3  
Alias name: mozillacert37.pem  
MD5: AB:57:A6:5B:7D:42:82:19:B5:D8:58:26:28:5E:FD:FF  
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:A1:2D  
Alias name: mozillacert4.pem  
MD5: 4F:EB:F1:F0:70:C2:80:63:5D:58:9F:DA:12:3C:A9:C4  
SHA256:  
OB: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:3F  
Alias name: mozillacert70.pem  
MD5: 5E:80:9E:84:5A:0E:65:0B:17:02:F3:55:18:2A:3E:D7  
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:C0  
Alias name: mozillacert88.pem  
MD5: 73:9F:4C:4B:73:5B:79:E9:FA:BA:1C:EF:6E:CB:D5:C9  
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:71  
Alias name: mozillacert134.pem  
MD5: FC:11:B8:D8:08:93:30:00:6D:23:F9:7E:EB:52:1E:02  
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:D1:22  
Alias name: mozillacert26.pem  
MD5: DC:32:C3:A7:6D:25:57:C7:68:09:9D:EA:2D:A9:A2:D1  
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:73  
Alias name: buypassclass2ca  
MD5: 46:A7:D2:FE:45:FB:64:5A:A8:59:90:9B:78:44:9B:29  
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:48  
Alias name: chunghwaepkirootca  
MD5: 1B:2E:00:CA:26:06:90:3D:AD:FE:6F:15:68:D3:6B:B3  
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:D5  
Alias name: verisignclass2g2ca  
MD5: 2D:BB:E5:25:D3:D1:65:82:3A:B7:0E:FA:E6:EB:E2:E1  
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:A1  
Alias name: mozillacert77.pem  
MD5: CD:68:B6:A7:C7:C4:CE:75:E0:1D:4F:57:44:61:92:09  
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:72:44  
Alias name: mozillacert123.pem  
MD5: C1:62:3E:23:C5:82:73:9C:03:59:4B:2B:E9:77:49:7F  
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:69:27  
Alias name: utndatacorpsgcc  
MD5: B3:A5:3E:77:21:6D:AC:4A:C0:C9:FB:D5:41:3D:CA:06  
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:B3:48  
Alias name: mozillacert15.pem  
MD5: 88:2C:8C:52:B8:A2:3C:F3:F7:BB:03:EA:AE:AC:42:0B  
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:CB
```

```
Alias name: digicertglobalrootca
MD5: 79:E4:A9:84:0D:7D:3A:96:D7:C0:4F:E2:43:4C:89:2E
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:C7:01:61
Alias name: mozillacert66.pem
MD5: 3D:41:29:CB:1E:AA:11:74:CD:5D:B0:62:AF:B0:43:5B
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:D5:98
Alias name: mozillacert112.pem
MD5: 37:41:49:1B:18:56:9A:26:F5:AD:C2:66:FB:40:A5:4C
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:A3:89
Alias name: utnuserfirstclientauthemailca
MD5: D7:34:3D:EF:1D:27:09:28:E1:31:02:5B:13:2B:DD:F7
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:AE
Alias name: verisignc2g1.pem
MD5: B3:9C:25:B1:C3:2E:32:53:80:15:30:9D:4D:02:77:3E
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:E9
Alias name: mozillacert55.pem
MD5: 74:9D:EA:60:24:C4:FD:22:53:3E:CC:3A:72:D9:29:4F
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:15:57
Alias name: mozillacert101.pem
MD5: DF:F2:80:73:CC:F1:E6:61:73:FC:F5:42:E9:C5:7C:EE
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:50
Alias name: mozillacert119.pem
MD5: 94:14:77:7E:3E:5E:FD:8F:30:BD:41:B0:CF:E7:D0:30
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:9E
Alias name: verisignc3g1.pem
MD5: EF:5A:F1:33:EF:F1:CD:BB:51:02:EE:12:14:4B:96:C4
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:14:03:05
Alias name: mozillacert44.pem
MD5: 72:E4:4A:87:E3:69:40:80:77:EA:BC:E3:F4:FF:F0:E1
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:A3
Alias name: mozillacert108.pem
MD5: 3E:45:52:15:09:51:92:E1:B7:5D:37:9F:B1:87:29:8A
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:99
Alias name: mozillacert95.pem
MD5: 3D:3B:18:9E:2C:64:5A:E8:D5:88:CE:0E:F9:37:C2:EC
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:B2:4D
Alias name: keynectisrootca
MD5: CC:4D:AE:FB:30:6B:D8:38:FE:50:EB:86:61:4B:D2:26
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:A3:32
Alias name: mozillacert141.pem
MD5: A9:23:75:9B:BA:49:36:6E:31:C2:DB:F2:E7:66:BA:87
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:D6:93:55:59:6B
Alias name: equifaxsecureglobalebusinessca1
MD5: 51:F0:2A:33:F1:F5:55:39:07:F2:16:7A:47:C7:5D:63
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:AE
Alias name: affirmtrustpremiumca
MD5: C4:5D:0E:48:B6:AC:28:30:4E:0A:BC:F9:38:16:87:57
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:9A
Alias name: baltimorecodeesigningca
MD5: 90:F5:28:49:56:D1:5D:2C:B0:53:D4:4B:EF:6F:90:22
```

```
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:46:87  
Alias name: mozillacert33.pem  
MD5: 22:2D:A6:01:EA:7C:0A:F7:F0:6C:56:43:3F:77:76:D3  
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:37  
Alias name: mozillacert0.pem  
MD5: CA:3D:D3:68:F1:03:5C:D0:32:FA:B8:2B:59:E8:5A:DB  
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:36  
Alias name: mozillacert84.pem  
MD5: 49:63:AE:27:F4:D5:95:3D:D8:DB:24:86:B8:9C:07:53  
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:FB  
Alias name: mozillacert130.pem  
MD5: 65:58:AB:15:AD:57:6C:1E:A8:A7:B5:69:AC:BF:FF:EB  
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:01:04  
Alias name: mozillacert148.pem  
MD5: 4C:56:41:E5:0D:BB:2B:E8:CA:A3:ED:18:08:AD:43:39  
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:D8:81:21:37  
Alias name: mozillacert22.pem  
MD5: 02:26:C3:01:5E:08:30:37:43:A9:D0:7D:CF:37:E6:BF  
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:D0:6C  
Alias name: verisignc1g1.pem  
MD5: 97:60:E8:57:5F:D3:50:47:E5:43:0C:94:36:8A:B0:62  
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:33:55  
Alias name: mozillacert7.pem  
MD5: 32:4A:4B:BB:C8:63:69:9B:BE:74:9A:C6:DD:1D:46:24  
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:B6:58  
Alias name: mozillacert73.pem  
MD5: D6:39:81:C6:52:7E:96:69:FC:FC:CA:66:ED:05:F2:96  
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:F5  
Alias name: mozillacert137.pem  
MD5: D3:D9:BD:AE:9F:AC:67:24:B3:C8:1B:52:E1:B9:A9:BD  
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:E8  
Alias name: swisssignsilverg2ca  
MD5: E0:06:A1:C9:7D:CF:C9:FC:0D:CO:56:75:96:D8:62:13  
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:D5  
Alias name: mozillacert11.pem  
MD5: 87:CE:0B:7B:2A:0E:49:00:E1:58:71:9B:37:A8:93:72  
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:C6:32:5C  
Alias name: mozillacert29.pem  
MD5: D3:F3:A6:16:C0:FA:6B:1D:59:B1:2D:96:4D:0E:11:2E  
SHA256:  
15:F0:BA:00:A9:AC:7A:F3:AC:88:4C:07:2B:10:11:A0:77:BD:77:C0:97:F4:01:64:B2:F8:59:8A:BD:83:86:0C  
Alias name: mozillacert62.pem  
MD5: EF:5A:F1:33:EF:F1:CD:BB:51:02:EE:12:14:4B:96:C4  
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:14:03:05  
Alias name: mozillacert126.pem  
MD5: 77:0D:19:B1:21:FD:00:42:9C:3E:0C:A5:DD:0B:02:8E  
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:4C  
Alias name: securetrustca  
MD5: DC:32:C3:A7:6D:25:57:C7:68:09:9D:EA:2D:A9:A2:D1
```

```
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:73  
Alias name: soneraaclassica  
MD5: 33:B7:84:F5:5F:27:D7:68:27:DE:14:DE:12:2A:ED:6F  
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:32:31:3D  
Alias name: mozillacert18.pem  
MD5: F1:6A:22:18:C9:CD:DF:CE:82:1D:1D:B7:78:5C:A9:A5  
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:A0  
Alias name: mozillacert51.pem  
MD5: 18:98:C0:D6:E9:3A:FC:F9:B0:F5:0C:F7:4B:01:44:17  
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:BB  
Alias name: mozillacert69.pem  
MD5: A6:B0:CD:85:80:DA:5C:50:34:A3:39:90:2F:55:67:73  
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:D4:53:1F  
Alias name: mozillacert115.pem  
MD5: 2B:9B:9E:E4:7B:6C:1F:00:72:1A:CC:C1:77:79:DF:6A  
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:17:05:25:52  
Alias name: verisignclass3g5ca  
MD5: CB:17:E4:31:67:3E:E2:09:FE:45:57:93:F3:0A:FA:1C  
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:DF  
Alias name: utnuserfirsthardwareca  
MD5: 4C:56:41:E5:0D:BB:2B:E8:CA:A3:ED:18:08:AD:43:39  
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:D8:81:21:37  
Alias name: addtrustqualifiedca  
MD5: 27:EC:39:47:CD:DA:5A:AF:E2:9A:01:65:21:A9:4C:BB  
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:D0:60:16  
Alias name: mozillacert40.pem  
MD5: 56:5F:AA:80:61:12:17:F6:67:21:E6:2B:6D:61:56:8E  
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:B3:40:8E  
Alias name: mozillacert58.pem  
MD5: 01:5E:D8:6B:BD:6F:3D:8E:A1:31:F8:12:E0:98:73:6A  
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:D7:66  
Alias name: verisignclass3g3ca  
MD5: CD:68:B6:A7:C7:C4:CE:75:E0:1D:4F:57:44:61:92:09  
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:72:44  
Alias name: mozillacert104.pem  
MD5: 55:5D:63:00:97:BD:6A:97:F5:67:AB:4B:FB:6E:63:15  
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:F7  
Alias name: mozillacert91.pem  
MD5: 30:C9:E7:1E:6B:E6:14:EB:65:B2:16:69:20:31:67:4D  
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:55:7D  
Alias name: thawtepessoalfreemailca  
MD5: 53:4B:1D:17:58:58:1A:30:A1:90:F8:6E:5C:F2:CF:65  
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:E1:84  
Alias name: certplusclass3ppprimaryca  
MD5: E1:4B:52:73:D7:1B:DB:93:30:E5:BD:E4:09:6E:BE:FB  
SHA256:  
CC:C8:94:89:37:1B:AD:11:1C:90:61:9B:EA:24:0A:2E:6D:AD:D9:9F:6E:1D:4D:41:E5:8E:D6:DE:3D:02:85  
Alias name: verisignc3g4.pem  
MD5: 3A:52:E1:E7:FD:6F:3A:E3:6F:F3:6F:99:1B:F9:22:41  
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:79
```

```
Alias name: swisssigngoldg2ca
MD5: 24:77:D9:A8:91:D1:3B:FA:88:2D:C2:FF:F8:CD:33:93
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:95
Alias name: mozillacert47.pem
MD5: ED:41:F5:8C:50:C5:2B:9C:73:E6:EE:6C:EB:C2:A8:26
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:C5:2A
Alias name: mozillacert80.pem
MD5: 64:B0:09:55:CF:B1:D5:99:E2:BE:13:AB:A6:5D:EA:4D
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:84:88:82:14:23
Alias name: mozillacert98.pem
MD5: 43:5E:88:D4:7D:1A:4A:7E:FD:84:2E:52:EB:01:D4:6F
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:76
Alias name: mozillacert144.pem
MD5: A3:EC:75:0F:2E:88:DF:FA:48:01:4E:0B:5C:48:6F:FB
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:D9:27
Alias name: starfieldclass2ca
MD5: 32:4A:4B:BB:C8:63:69:9B:BE:74:9A:C6:DD:1D:46:24
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:B6:58
Alias name: mozillacert36.pem
MD5: F0:96:B6:2F:C5:10:D5:67:8E:83:25:32:E8:5E:2E:E5
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:D0
Alias name: mozillacert3.pem
MD5: 39:16:AA:B9:6A:41:E1:14:69:DF:9E:6C:3B:72:DC:B6
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:03:12
Alias name: globalsignr2ca
MD5: 94:14:77:7E:3E:5E:FD:8F:30:BD:41:B0:CF:E7:D0:30
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:9E
Alias name: mozillacert87.pem
MD5: 6C:39:7D:A4:0E:55:59:B2:3F:D6:41:B1:12:50:DE:43
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:F6
Alias name: mozillacert133.pem
MD5: D6:ED:3C:CA:E2:66:0F:AF:10:43:0D:77:9B:04:09:BF
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:BD
Alias name: mozillacert25.pem
MD5: CB:17:E4:31:67:3E:E2:09:FE:45:57:93:F3:0A:FA:1C
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:DF
Alias name: verisignclass1g2ca
MD5: DB:23:3D:F9:69:FA:4B:B9:95:80:44:73:5E:7D:41:83
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:7F
Alias name: mozillacert76.pem
MD5: 82:92:BA:5B:EF:CD:8A:6F:A6:3D:55:F9:84:F6:D6: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:A7
Alias name: mozillacert122.pem
MD5: 1D:35:54:04:85:78:B0:3F:42:42:4D:BF:20:73:0A:3F
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:F2
Alias name: mozillacert14.pem
MD5: D4:74:DE:57:5C:39:B2:D3:9C:85:83:C5:C0:65:49:8A
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:CF
Alias name: equifaxsecureca
MD5: 67:CB:9D:C0:13:24:8A:82:9B:B2:17:1E:D1:1B:EC:D4
```

```
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:F1:78  
Alias name: mozillacert65.pem  
MD5: A2:6F:53:B7:EE:40:DB:4A:68:E7:FA:18:D9:10:4B:72  
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:73:01:9A  
Alias name: mozillacert111.pem  
MD5: F9:03:7E:CF:E6:9E:3C:73:7A:2A:90:07:69:FF:2B:96  
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:1B  
Alias name: certumtrustednetworkca  
MD5: D5:E9:81:40:C5:18:69:FC:46:2C:89:75:62:0F:AA:78  
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:C4:40:8E  
Alias name: mozillacert129.pem  
MD5: 92:65:58:8B:A2:1A:31:72:73:68:5C:B4:A5:7A:07:48  
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:B9:12  
Alias name: mozillacert54.pem  
MD5: B5:E8:34:36:C9:10:44:58:48:70:6D:2E:83:D4:B8:05  
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:D4  
Alias name: mozillacert100.pem  
MD5: CD:E0:25:69:8D:47:AC:9C:89:35:90:F7:FD:51:3D:2F  
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:C1  
Alias name: mozillacert118.pem  
MD5: 8F:5D:77:06:27:C4:98:3C:5B:93:78:E7:D7:7D:9B:CC  
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:07  
Alias name: gd-class2-root.pem  
MD5: 91:DE:06:25:AB:DA:FD:32:17:0C:BB:25:17:2A:84:67  
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:E4  
Alias name: mozillacert151.pem  
MD5: 86:38:6D:5E:49:63:6C:85:5C:DB:6D:DC:94:B7:D0:F7  
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:F6:67  
Alias name: thawteprimaryrootcag3  
MD5: FB:1B:5D:43:8A:94:CD:44:C6:76:F2:43:4B:47:E7:31  
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:D3:4C  
Alias name: quovadisrootca  
MD5: 27:DE:36:FE:72:B7:00:03:00:9D:F4:F0:1E:6C:04:24  
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:88:73  
Alias name: thawteprimaryrootcag2  
MD5: 74:9D:EA:60:24:C4:FD:22:53:3E:CC:3A:72:D9:29:4F  
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:15:57  
Alias name: deprecateditsecca  
MD5: A5:96:0C:F6:B5:AB:27:E5:01:C6:00:88:9E:60:33:E5  
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:CB  
Alias name: entrustrootcag2  
MD5: 4B:E2:C9:91:96:65:0C:F4:0E:5A:93:92:A0:0A:FE:B2  
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:F3:39  
Alias name: mozillacert43.pem  
MD5: 40:01:25:06:8D:21:43:6A:0E:43:00:9C:E7:43:F3:D5  
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:D2  
Alias name: mozillacert107.pem  
MD5: BE:EC:11:93:9A:F5:69:21:BC:D7:C1:C0:67:89:CC:2A  
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:CE
```

```
Alias name: trustcenterclass4caii
MD5: 9D:FB:F9:AC:ED:89:33:22:F4:28:48:83:25:23:5B:E0
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:32
Alias name: mozillacert94.pem
MD5: 46:A7:D2:FE:45:FB:64:5A:A8:59:90:9B:78:44:9B:29
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:48
Alias name: mozillacert140.pem
MD5: 5E:39:7B:DD:F8:BA:EC:82:E9:AC:62:BA:0C:54:00:2B
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:86
Alias name: ttelesecglobalrootclass3ca
MD5: CA:FB:40:A8:4E:39:92:8A:1D:FE:8E:2F:C4:27:EA:EF
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:BD
Alias name: amzninternalcorpca
MD5: 7B:0E:9D:67:A9:3A:88:DD:BA:81:8D:A9:3C:74:AA:BB
SHA256:
01:29:04:6C:60:EF:5C:51:60:D3:9F:A2:3A:1D:0C:52:0A:AF:DA:4F:17:87:95:AA:66:82:01:9F:76:C9:11:DC
Alias name: starfieldservicesrootg2ca
MD5: 17:35:74:AF:7B:61:1C:EB:F4:F9:3C:E2:EE:40:F9:A2
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:B5
Alias name: mozillacert32.pem
MD5: 0C:7F:DD:6A:F4:2A:B9:C8:9B:BD:20:7E:A9:DB:5C:37
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:89:32
Alias name: mozillacert83.pem
MD5: 2C:8C:17:5E:B1:54:AB:93:17:B5:36:5A:DB:D1:C6:F2
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:63:0E
Alias name: verisignroot.pem
MD5: 8E:AD:B5:01:AA:4D:81:E4:8C:1D:D1:E1:14:00:95:19
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:3C
Alias name: mozillacert147.pem
MD5: B3:A5:3E:77:21:6D:AC:4A:C0:C9:FB:D5:41:3D:CA:06
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:B3:48
Alias name: camerfirmachambersca
MD5: 5E:80:9E:84:5A:0E:65:0B:17:02:F3:55:18:2A:3E:D7
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:C0
Alias name: mozillacert21.pem
MD5: E0:06:A1:C9:7D:CF:C9:FC:0D:CO:56:75:96:D8:62:13
SHA256:
BE:6C:4D:A2:BB:B9:BA:59:B6:F3:93:97:68:37:42:46:C3:CO:05:99:3F:A9:8F:02:0D:1D:ED:BE:D4:8A:81:D5
Alias name: mozillacert39.pem
MD5: CE:78:33:5C:59:78:01:6E:18:EA:B9:36:A0:B9:2E:23
SHA256:
E6:B8:F8:76:64:85:F8:07:AE:7F:8D:AC:16:70:46:1F:07:CO:A1:3E:EF:3A:1F:F7:17:53:8D:7A:BA:D3:91:B4
Alias name: mozillacert6.pem
MD5: 91:DE:06:25:AB:DA:FD:32:17:0C:BB:25:17:2A:84:67
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:E4
Alias name: verisignuniversalrootca
MD5: 8E:AD:B5:01:AA:4D:81:E4:8C:1D:D1:E1:14:00:95:19
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:3C
Alias name: mozillacert72.pem
MD5: 80:3A:BC:22:C1:E6:FB:8D:9B:3B:27:4A:32:1B:9A:01
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:DA
Alias name: geotrustuniversalca
MD5: 92:65:58:8B:A2:1A:31:72:73:68:5C:B4:A5:7A:07:48
```

```
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:B9:12  
Alias name: mozillacert136.pem  
MD5: 49:79:04:B0:EB:87:19:AC:47:B0:BC:11:51:9B:74:D0  
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:F4  
Alias name: mozillacert10.pem  
MD5: F8:38:7C:77:88:DF:2C:16:68:2E:C2:E2:52:4B:B8:F9  
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:D9:2E  
Alias name: mozillacert28.pem  
MD5: 5C:48:DC:F7:42:72:EC:56:94:6D:1C:CC:71:35:80:75  
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:66  
Alias name: affirmtrustnetworkingca  
MD5: 42:65:CA:BE:01:9A:9A:4C:A9:8C:41:49:CD:C0:D5:7F  
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:B4:1B  
Alias name: mozillacert61.pem  
MD5: 42:81:A0:E2:1C:E3:55:10:DE:55:89:42:65:96:22:E6  
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:43:4A  
Alias name: mozillacert79.pem  
MD5: C4:5D:0E:48:B6:AC:28:30:4E:0A:BC:F9:38:16:87:57  
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:9A  
Alias name: affirmtrustcommercialca  
MD5: 82:92:BA:5B:EF:CD:8A:6F:A6:3D:55:F9:84:F6:D6: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:A7  
Alias name: mozillacert125.pem  
MD5: D6:A5:C3:ED:5D:DD:3E:00:C1:3D:87:92:1F:1D:3F:E4  
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:4C  
Alias name: mozillacert17.pem  
MD5: 21:D8:4C:82:2B:99:09:33:A2:EB:14:24:8D:8E:5F:E8  
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:40  
Alias name: mozillacert50.pem  
MD5: 2C:20:26:9D:CB:1A:4A:00:85:B5:B7:5A:AE:C2:01:37  
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:B2  
Alias name: mozillacert68.pem  
MD5: 73:3A:74:7A:EC:BB:A3:96:A6:C2:E4:E2:C8:9B:C0:C3  
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:EF  
Alias name: starfieldrootg2ca  
MD5: D6:39:81:C6:52:7E:96:69:FC:FC:CA:66:ED:05:F2:96  
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:F5  
Alias name: mozillacert114.pem  
MD5: B8:A1:03:63:B0:BD:21:71:70:8A:6F:13:3A:BB:79:49  
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:F2:3C  
Alias name: buypassclass3ca  
MD5: 3D:3B:18:9E:2C:64:5A:E8:D5:88:CE:0E:F9:37:C2:EC  
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:B2:4D  
Alias name: mozillacert57.pem  
MD5: A8:0D:6F:39:78:B9:43:6D:77:42:6D:98:5A:CC:23:CA  
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:B2  
Alias name: verisignc2g3.pem  
MD5: F8:BE:C4:63:22:C9:A8:46:74:8B:B8:1D:1E:4A:2B:F6  
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:BB
```

```
Alias name: verisignclass2g3ca
MD5: F8:BE:C4:63:22:C9:A8:46:74:8B:B8:1D:1E:4A:2B:F6
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:BB
Alias name: mozillacert103.pem
MD5: E6:24:E9:12:01:AE:0C:DE:8E:85:C4:CE:A3:12:DD:EC
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:B0
Alias name: mozillacert90.pem
MD5: 69:C1:0D:4F:07:A3:1B:C3:FE:56:3D:04:BC:11:F6:A6
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:D7:60:66
Alias name: verisignc3g3.pem
MD5: CD:68:B6:A7:C7:C4:CE:75:E0:1D:4F:57:44:61:92:09
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:72:44
Alias name: mozillacert46.pem
MD5: AA:8E:5D:D9:F8:DB:0A:58:B7:8D:26:87:6C:82:35:55
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:E9:39
Alias name: godaddyclass2ca
MD5: 91:DE:06:25:AB:DA:FD:32:17:0C:BB:25:17:2A:84:67
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:E4
Alias name: verisignc4g3.pem
MD5: DB:C8:F2:27:2E:B1:EA:6A:29:23:5D:FE:56:3E:33:DF
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:96:05:61:79:15:06
Alias name: mozillacert97.pem
MD5: A2:33:9B:4C:74:78:73:D4:6C:E7:C1:F3:8D:CB:5C:E9
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:A8:88:63:8B
Alias name: mozillacert143.pem
MD5: F1:BC:63:6A:54:E0:B5:27:F5:CD:E7:1A:E3:4D:6E:4A
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:6C
Alias name: mozillacert35.pem
MD5: 3F:45:96:39:E2:50:87:F7:BB:FE:98:0C:3C:20:98:E6
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:CF
Alias name: mozillacert2.pem
MD5: 3A:52:E1:E7:FD:6F:3A:E3:6F:F3:6F:99:1B:F9:22:41
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:79
Alias name: utnuserfirstobjectca
MD5: A7:F2:E4:16:06:41:11:50:30:6B:9C:E3:B4:9C:B0:C9
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:E5:8F
Alias name: mozillacert86.pem
MD5: 10:FC:63:5D:F6:26:3E:0D:F3:25:BE:5F:79:CD:67:67
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:70
Alias name: mozillacert132.pem
MD5: 14:F1:08:AD:9D:FA:64:E2:89:E7:1C:CF:A8:AD:7D:5E
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:E3
Alias name: addtrustclass1ca
MD5: 1E:42:95:02:33:92:6B:B9:5F:C0:7F:DA:D6:B2:4B:FC
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:A7
Alias name: mozillacert24.pem
MD5: 7C:A5:0F:F8:5B:9A:7D:6D:30:AE:54:5A:E3:42:A2:8A
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:B6:71:6F
Alias name: verisignc1g3.pem
MD5: B1:47:BC:18:57:D1:18:A0:78:2D:EC:71:E8:2A:95:73
```

```
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:65  
Alias name: mozillacert9.pem  
MD5: 37:85:44:53:32:45:1F:20:F0:F3:95:E1:25:C4:43:4E  
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:B3  
Alias name: amzninternalrootca  
MD5: 08:09:73:AC:E0:78:41:7C:0A:26:33:51:E8:CF:E6:60  
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:AA  
Alias name: mozillacert75.pem  
MD5: 67:CB:9D:C0:13:24:8A:82:9B:B2:17:1E:D1:1B:EC:D4  
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:F1:78  
Alias name: entrustevca  
MD5: D6:A5:C3:ED:5D:DD:3E:00:C1:3D:87:92:1F:1D:3F:E4  
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:4C  
Alias name: secomscrootca2  
MD5: 6C:39:7D:A4:0E:55:59:B2:3F:D6:41:B1:12:50:DE:43  
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:F6  
Alias name: camerfirmachambersignca  
MD5: 9E:80:FF:78:01:0C:2E:C1:36:BD:FE:96:90:6E:08:F3  
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:CA  
Alias name: secomscrootca1  
MD5: F1:BC:63:6A:54:E0:B5:27:F5:CD:E7:1A:E3:4D:6E:4A  
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:6C  
Alias name: mozillacert121.pem  
MD5: 1E:42:95:02:33:92:6B:B9:5F:C0:7F:DA:D6:B2:4B:FC  
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:A7  
Alias name: mozillacert139.pem  
MD5: 27:DE:36:FE:72:B7:00:03:00:9D:F4:F0:1E:6C:04:24  
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:88:73  
Alias name: mozillacert13.pem  
MD5: C5:A1:B7:FF:73:DD:D6:D7:34:32:18:DF:FC:3C:AD:88  
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:98  
Alias name: mozillacert64.pem  
MD5: 06:9F:69:79:16:66:90:02:1B:8C:8C:A2:C3:07:6F:3A  
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:72  
Alias name: mozillacert110.pem  
MD5: D0:A0:5A:EE:05:B6:09:94:21:A1:7D:F1:B2:29:82:02  
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:D1:04:38:41:13  
Alias name: mozillacert128.pem  
MD5: 0E:40:A7:6C:DE:03:5D:8F:D1:0F:E4:D1:8D:F9:6C:A9  
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:85  
Alias name: entrust2048ca  
MD5: EE:29:31:BC:32:7E:9A:E6:E8:B5:F7:51:B4:34:71:90  
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:B1:77  
Alias name: mozillacert53.pem  
MD5: 7E:23:4E:5B:A7:A5:B4:25:E9:00:07:74:11:62:AE:D6  
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:19  
Alias name: mozillacert117.pem  
MD5: AC:B6:94:A5:9C:17:E0:D7:91:52:9B:B1:97:06:A6:E4  
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:EB
```

```
Alias name: mozillacert150.pem
MD5: C5:E6:7B:BF:06:D0:4F:43:ED:C4:7A:65:8A:FB:6B:19
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:ED
Alias name: thawteserverca
MD5: EE:FE:61:69:65:6E:F8:9C:C6:2A:F4:D7:2B:63:EF:A2
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:10:55:31:6B
Alias name: secomvalicertclass1ca
MD5: 65:58:AB:15:AD:57:6C:1E:A8:A7:B5:69:AC:BF:FF:EB
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:01:04
Alias name: mozillacert42.pem
MD5: 74:01:4A:91:B1:08:C4:58:CE:47:CD:F0:DD:11:53:08
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:D3
Alias name: verisignc2g6.pem
MD5: 7D:0B:83:E5:FB:7C:AD:07:4F:20:A9:B5:DF:63:ED:79
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:F0
Alias name: godaddyrootg2ca
MD5: 80:3A:BC:22:C1:E6:FB:8D:9B:3B:27:4A:32:1B:9A:01
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:DA
Alias name: gtecybertrustglobalca
MD5: CA:3D:D3:68:F1:03:5C:D0:32:FA:B8:2B:59:E8:5A:DB
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:36
Alias name: mozillacert106.pem
MD5: 7B:30:34:9F:DD:0A:4B:6B:35:CA:31:51:28:5D:AE:EC
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:B2:58:01:5D
Alias name: equifaxsecurebusinessca1
MD5: 14:C0:08:E5:A3:85:03:A3:BE:78:E9:67:4F:27:CA:EE
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:F5:96
Alias name: mozillacert93.pem
MD5: 78:4B:FB:9E:64:82:0A:D3:B8:4C:62:F3:64:F2:90:64
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:59:95
Alias name: quovaldisrootca3
MD5: 31:85:3C:62:94:97:63:B9:AA:FD:89:4E:AF:6F:E0:CF
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:35
Alias name: quovaldisrootca2
MD5: 5E:39:7B:DD:F8:BA:EC:82:E9:AC:62:BA:0C:54:00:2B
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:86
Alias name: soneraaclass2ca
MD5: A3:EC:75:0F:2E:88:DF:FA:48:01:4E:0B:5C:48:6F:FB
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:D9:27
Alias name: mozillacert31.pem
MD5: 7C:62:FF:74:9D:31:53:5E:68:4A:D5:78:AA:1E:BF:23
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:C7
Alias name: mozillacert49.pem
MD5: DF:3C:73:59:81:E7:39:50:81:04:4C:34:A2:CB:B3:7B
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:24
Alias name: mozillacert82.pem
MD5: 7F:30:78:8C:03:E3:CA:C9:0A:E2:C9:EA:1E:AA:55:1A
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:17
Alias name: mozillacert146.pem
MD5: 91:F4:03:55:20:A1:F8:63:2C:62:DE:AC:FB:61:1C:8E
```

```
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:E7  
Alias name: baltimorecybertrustca  
MD5: AC:B6:94:A5:9C:17:E0:D7:91:52:9B:B1:97:06:A6:E4  
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:EB  
Alias name: mozillacert20.pem  
MD5: 24:77:D9:A8:91:D1:3B:FA:88:2D:C2:FF:F8:CD:33:93  
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:95  
Alias name: mozillacert38.pem  
MD5: 93:2A:3E:F6:FD:23:69:0D:71:20:D4:2B:47:99:2B:A6  
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:E3:43:08:5C  
Alias name: mozillacert5.pem  
MD5: A1:0B:44:B3:CA:10:D8:00:6E:9D:0F:D8:0F:92:0A:D1  
SHA256:  
CE:CD:DC:90:50:99:D8:DA:DF:C5:B1:D2:09:B7:37:CB:E2:C1:8C:FB:2C:10:CO:FF:0B:CF:0D:32:86:FC:1A:A2  
Alias name: mozillacert71.pem  
MD5: 9E:80:FF:78:01:0C:2E:C1:36:BD:FE:96:90:6E:08:F3  
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:CA  
Alias name: verisignclass3g4ca  
MD5: 3A:52:E1:E7:FD:6F:3A:E3:6F:F3:6F:99:1B:F9:22:41  
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:79  
Alias name: mozillacert89.pem  
MD5: DB:C8:F2:27:2E:B1:EA:6A:29:23:5D:FE:56:3E:33:DF  
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:96:05:61:79:15:06  
Alias name: mozillacert135.pem  
MD5: 2C:8F:9F:66:1D:18:90:B1:47:26:9D:8E:86:82:8C:A9  
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:D8:14:36:24  
Alias name: camerfirmachamberscommercecea  
MD5: B0:01:EE:14:D9:AF:29:18:94:76:8E:F1:69:33:2A:84  
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:C3  
Alias name: mozillacert27.pem  
MD5: CF:F4:27:0D:D4:ED:DC:65:16:49:6D:3D:DA:BF:6E:DE  
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:69  
Alias name: verisignc1g6.pem  
MD5: 2F:A8:B4:DA:F6:64:4B:1E:82:F9:46:3D:54:1A:7C:B0  
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:B9  
Alias name: verisignclass3g2ca  
MD5: A2:33:9B:4C:74:78:73:D4:6C:E7:C1:F3:8D:CB:5C:E9  
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:A8:88:63:8B  
Alias name: mozillacert60.pem  
MD5: B7:52:74:E2:92:B4:80:93:F2:75:E4:CC:D7:F2:EA:26  
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:42:16:12  
Alias name: mozillacert78.pem  
MD5: 42:65:CA:BE:01:9A:9A:4C:A9:8C:41:49:CD:C0:D5:7F  
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:B4:1B  
Alias name: gd_bundle-g2.pem  
MD5: 96:C2:50:31:BC:0D:C3:5C:FB:A7:23:73:1E:1B:41:40  
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:A6  
Alias name: certumca  
MD5: 2C:8F:9F:66:1D:18:90:B1:47:26:9D:8E:86:82:8C:A9  
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:D8:14:36:24
```

```
Alias name: deutschetelekomrootca2
MD5: 74:01:4A:91:B1:08:C4:58:CE:47:CD:F0:DD:11:53:08
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:D3
Alias name: mozillacert124.pem
MD5: 27:EC:39:47:CD:DA:5A:AF:E2:9A:01:65:21:A9:4C:BB
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:DO:60:16
Alias name: mozillacert16.pem
MD5: 41:03:52:DC:0F:F7:50:1B:16:F0:02:8E:BA:6F:45:C5
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:D6:77:07:39
Alias name: secomevrootca1
MD5: 22:2D:A6:01:EA:7C:0A:F7:F0:6C:56:43:3F:77:76:D3
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:37
Alias name: mozillacert67.pem
MD5: C5:DF:B8:49:CA:05:13:55:EE:2D:BA:1A:C3:3E:B0:28
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:C5:3B
Alias name: globalsignr3ca
MD5: C5:DF:B8:49:CA:05:13:55:EE:2D:BA:1A:C3:3E:B0:28
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:C5:3B
Alias name: mozillacert113.pem
MD5: EE:29:31:BC:32:7E:9A:E6:E8:B5:F7:51:B4:34:71:90
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:B1:77
Alias name: gdroot-g2.pem
MD5: 80:3A:BC:22:C1:E6:FB:8D:9B:3B:27:4A:32:1B:9A:01
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:DA
Alias name: aolrootca2
MD5: D6:ED:3C:CA:E2:66:0F:AF:10:43:0D:77:9B:04:09:BF
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:BD
Alias name: trustcenteruniversalcai
MD5: 45:E1:A5:72:C5:A9:36:64:40:9E:F5:E4:58:84:67:8C
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:E7
Alias name: aolrootca1
MD5: 14:F1:08:AD:9D:FA:64:E2:89:E7:1C:CF:A8:AD:7D:5E
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:E3
Alias name: verisignc2g2.pem
MD5: 2D:BB:E5:25:D3:D1:65:82:3A:B7:0E:FA:E6:EB:E2:E1
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:A1
Alias name: mozillacert56.pem
MD5: FB:1B:5D:43:8A:94:CD:44:C6:76:F2:43:4B:47:E7:31
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:D3:4C
Alias name: verisignclass1g3ca
MD5: B1:47:BC:18:57:D1:18:A0:78:2D:EC:71:E8:2A:95:73
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:65
Alias name: mozillacert102.pem
MD5: AA:C6:43:2C:5E:2D:CD:C4:34:C0:50:4F:11:02:4F:B6
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:D3:48:81
Alias name: addtrustexternalca
MD5: 1D:35:54:04:85:78:B0:3F:42:42:4D:BF:20:73:0A:3F
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:F2
Alias name: verisignc3g2.pem
MD5: A2:33:9B:4C:74:78:73:D4:6C:E7:C1:F3:8D:CB:5C:E9
```

```
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:A8:88:63:8B  
Alias name: verisignclass3ca  
MD5: EF:5A:F1:33:EF:F1:CD:BB:51:02:EE:12:14:4B:96:C4  
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:14:03:05  
Alias name: mozillacert45.pem  
MD5: 1B:2E:00:CA:26:06:90:3D:AD:FE:6F:15:68:D3:6B:B3  
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:D5  
Alias name: verisignc4g2.pem  
MD5: 26:6D:2C:19:98:B6:70:68:38:50:54:19:EC:90:34:60  
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:C7:01:19:61  
Alias name: digicertassuredidrootca  
MD5: 87:CE:0B:7B:2A:0E:49:00:E1:58:71:9B:37:A8:93:72  
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:C6:32:5C  
Alias name: verisignclass1ca  
MD5: 86:AC:DE:2B:C5:6D:C3:D9:8C:28:88:D3:8D:16:13:1E  
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:B8:84:2C  
Alias name: mozillacert109.pem  
MD5: 26:01:FB:D8:27:A7:17:9A:45:54:38:1A:43:01:3B:03  
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:F8:07:03  
Alias name: thawtepremiumserverca  
MD5: A6:6B:60:90:23:9B:3F:2D:BB:98:6F:D6:A7:19:0D:46  
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:E9  
Alias name: verisigntsaca  
MD5: F2:89:95:6E:4D:05:F0:F1:A7:21:55:7D:46:11:BA:47  
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:9D  
Alias name: mozillacert96.pem  
MD5: CA:FB:40:A8:4E:39:92:8A:1D:FE:8E:2F:C4:27:EA:EF  
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:BD  
Alias name: mozillacert142.pem  
MD5: 31:85:3C:62:94:97:63:B9:AA:FD:89:4E:AF:6F:E0:CF  
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:35  
Alias name: thawteprimaryrootca  
MD5: 8C:CA:DC:0B:22:CE:F5:BE:72:AC:41:1A:11:A8:D8:12  
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:9F  
Alias name: mozillacert34.pem  
MD5: BC:6C:51:33:A7:E9:D3:66:63:54:15:72:1B:21:92:93  
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:F5  
Alias name: mozillacert1.pem  
MD5: C5:70:C4:A2:ED:53:78:0C:C8:10:53:81:64:CB:D0:1D  
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:E9  
Alias name: xrampglobalca  
MD5: A1:0B:44:B3:CA:10:D8:00:6E:9D:0F:D8:0F:92:0A:D1  
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:A2  
Alias name: mozillacert85.pem  
MD5: AA:08:8F:F6:F9:7B:B7:F2:B1:A7:1E:9B:EA:EA:BD:79  
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:44  
Alias name: valicertclass2ca  
MD5: A9:23:75:9B:BA:49:36:6E:31:C2:DB:F2:E7:66:BA:87  
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:D6:93:55:59:6B
```

```
Alias name: mozillacert131.pem
MD5: 34:FC:B8:D0:36:DB:9E:14:B3:C2:F2:DB:8F:E4:94:C7
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:0B
Alias name: mozillacert149.pem
MD5: B0:01:EE:14:D9:AF:29:18:94:76:8E:F1:69:33:2A:84
SHA256:
OC: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:C3
Alias name: geotrustprimaryca
MD5: 02:26:C3:01:5E:08:30:37:43:A9:D0:7D:CF:37:E6:BF
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:D0:6C
Alias name: mozillacert23.pem
MD5: 8C:CA:DC:0B:22:CE:F5:BE:72:AC:41:1A:11:A8:D8:12
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:9F
Alias name: verisignc1g2.pem
MD5: DB:23:3D:F9:69:FA:4B:B9:95:80:44:73:5E:7D:41:83
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:7F
Alias name: mozillacert8.pem
MD5: 22:4D:8F:8A:FC:F7:35:C2:BB:57:34:90:7B:8B:22:16
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:EA
Alias name: mozillacert74.pem
MD5: 17:35:74:AF:7B:61:1C:EB:F4:F9:3C:E2:EE:40:F9:A2
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:B5
Alias name: mozillacert120.pem
MD5: 64:9C:EF:2E:44:FC:C6:8F:52:07:D0:51:73:8F:CB:3D
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:F6:73
Alias name: geotrustglobalca
MD5: F7:75:AB:29:FB:51:4E:B7:77:5E:FF:05:3C:99:8E:F5
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:A7:3A
Alias name: mozillacert138.pem
MD5: 91:1B:3F:6E:CD:9E:AB:EE:07:FE:1F:71:D2:B3:61:27
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:69
Alias name: mozillacert12.pem
MD5: 79:E4:A9:84:0D:7D:3A:96:D7:CO:4F:E2:43:4C:89:2E
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:C7:01:61
Alias name: comodoaaaca
MD5: 49:79:04:B0:EB:87:19:AC:47:BO:BC:11:51:9B:74:D0
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:F4
Alias name: mozillacert63.pem
MD5: F8:49:F4:03:BC:44:2D:83:BE:48:69:7D:29:64:FC:B1
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:F3:78
Alias name: certplusclass2primaryca
MD5: 88:2C:8C:52:B8:A2:3C:F3:F7:BB:03:EA:AE:AC:42:0B
SHA256:
OF: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:CB
Alias name: mozillacert127.pem
MD5: F7:75:AB:29:FB:51:4E:B7:77:5E:FF:05:3C:99:8E:F5
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:A7:3A
Alias name: ttelesecglobalrootclass2ca
MD5: 2B:9B:9E:E4:7B:6C:1F:00:72:1A:CC:C1:77:79:DF:6A
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:17:05:25:52
Alias name: mozillacert19.pem
MD5: 37:A5:6E:D4:B1:25:84:97:B7:FD:56:15:7A:F9:A2:00
```

```
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:E6  
Alias name: digicerthighassuranceevrootca  
MD5: D4:74:DE:57:5C:39:B2:D3:9C:85:83:C5:C0:65:49:8A  
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:CF  
Alias name: amzninternalinfoseccag3  
MD5: E9:34:94:02:BA:BB:31:6B:22:E6:2B:A9:C4:F0:26:04  
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:68  
Alias name: mozillacert52.pem  
MD5: 21:BC:82:AB:49:C4:13:3B:4B:B2:2B:5C:6B:90:9C:19  
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:A7  
Alias name: mozillacert116.pem  
MD5: AE:B9:C4:32:4B:AC:7F:5D:66:CC:77:94:BB:2A:77:56  
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:A4:74  
Alias name: globalsignca  
MD5: 3E:45:52:15:09:51:92:E1:B7:5D:37:9F:B1:87:29:8A  
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:99  
Alias name: mozillacert41.pem  
MD5: 45:E1:A5:72:C5:A9:36:64:40:9E:F5:E4:58:84:67:8C  
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:E7  
Alias name: mozillacert59.pem  
MD5: 8E:AD:B5:01:AA:4D:81:E4:8C:1D:D1:E1:14:00:95:19  
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:3C  
Alias name: mozillacert105.pem  
MD5: 5B:04:69:EC:A5:83:94:63:18:A7:86:D0:E4:F2:6E:19  
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:38:41  
Alias name: trustcenterclass2caii  
MD5: CE:78:33:5C:59:78:01:6E:18:EA:B9:36:A0:B9:2E:23  
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:B4  
Alias name: mozillacert92.pem  
MD5: C9:3B:0D:84:41:FC:A4:76:79:23:08:57:DE:10:19:16  
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:A9:11  
Alias name: verisignc3g5.pem  
MD5: CB:17:E4:31:67:3E:E2:09:FE:45:57:93:F3:0A:FA:1C  
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:DF  
Alias name: geotrustprimarycag3  
MD5: B5:E8:34:36:C9:10:44:58:48:70:6D:2E:83:D4:B8:05  
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:D4  
Alias name: geotrustprimarycag2  
MD5: 01:5E:D8:6B:BD:6F:3D:8E:A1:31:F8:12:E0:98:73:6A  
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:D7:66  
Alias name: mozillacert30.pem  
MD5: 15:AC:A5:C2:92:2D:79:BC:E8:7F:CB:67:ED:02:CF:36  
SHA256:  
A7:12:72:AE:AA:A3:CF:E8:72:7F:B3:9F:0F:B3:D1:E5:42:6E:90:60:B0:6E:E6:F1:3E:9A:3C:58:33:CD:43  
Alias name: affirmtrustpremiumeccca  
MD5: 64:B0:09:55:CF:B1:D5:99:E2:BE:13:AB:A6:5D:EA:4D  
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:84:88:82:14:23  
Alias name: mozillacert48.pem  
MD5: B8:08:9A:F0:03:CC:1B:0D:C8:6C:0B:76:A1:75:64:23
```

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:38

Use o AWS WAF para proteger sua API do Amazon API Gateway de explorações comuns na web

O AWS WAF é um firewall para aplicativos web que ajuda a proteger aplicativos web e APIs de ataques, possibilitando que você configure um conjunto de regras (conhecido como lista de controle de acesso da web ou ACL da web) que permite, bloqueia ou calcula solicitações da web com base nas regras de segurança da web personalizáveis e nas condições que você definir. Para obter mais informações, consulte [Como o AWS WAF funciona](#).

Você pode usar o AWS WAF para proteger sua API do API Gateway de violações comuns na web, como ataques de injeção SQL e cross-site scripting (XSS), que poderiam afetar a disponibilidade e o desempenho da API, comprometendo a segurança ou consumindo recursos em excesso. Por exemplo, você pode criar regras para permitir ou bloquear solicitações de intervalos de endereços IP ou blocos CIDR especificados originadas de um determinado país ou região, que contenha código SQL mal-intencionado ou scripts mal-intencionados. 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 8 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

O AWS WAF 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 [recursos \(p. 362\)](#), [políticas do IAM \(p. 378\)](#), [autorizadores do Lambda \(p. 396\)](#) e [autorizadores do Amazon Cognito \(p. 414\)](#). Por exemplo, se o AWS WAF bloqueia o acesso de um bloco CIDR que uma política de recurso permite, o AWS WAF tem precedência, e a política de recurso não é avaliada.

Para habilitar o AWS WAF para sua API, será necessário:

1. Usar o console do AWS WAF, o AWS SDK ou a CLI para criar uma ACL regional da web que contenha a combinação desejada de regras gerenciadas do AWS WAF e suas próprias regras personalizadas. Para obter mais informações, consulte [Conceitos básicos do AWS WAF](#) e [Criar e configurar uma lista de controle de acesso à web \(ACL da web\)](#).

Important

O API Gateway requer uma ACL regional da web.

2. Associe a ACL regional da web do AWS WAF a um estágio de API. A associação pode ser feita com o uso do console do AWS WAF, o AWS SDK ou a CLI, ou pelo console do API Gateway, o AWS SDK ou a CLI.

Para associar uma ACL regional da web do AWS WAF a um estágio de API do API Gateway usando o console do API Gateway

Para usar o console do API Gateway visando associar uma ACL regional da web do AWS WAF a um estágio de API do API Gateway existente, siga estas etapas:

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação de APIs, selecione a API e os Stages (Estágios).
3. No painel Stages (Estágios), escolha o nome do estágio.
4. No painel Stage Editor (Editor de estágio), selecione a guia Settings (Configurações).
5. Para associar uma ACL regional da web com o estágio de API:
 - Na lista suspensa de ACL da web do AWS WAF, selecione a ACL regional da web que você deseja associar a esse estágio.

Note

Se a ACL da web de que você precisa ainda não existir, selecione Create WebACL (Criar WebACL) e, em seguida, escolha Go to AWS WAF (Acessar AWS WAF) para abrir o console do WAF em uma nova guia do navegador e criar uma ACL regional da web. Em seguida, retorne ao console do API Gateway para associar a ACL da web ao estágio.

6. Selecione Save Changes (Salvar alterações).

Associar uma ACL regional da web do AWS WAF a um estágio de API do API Gateway usando um estágio de API do AWS CLI

Para usar a AWS CLI visando associar uma ACL regional da web do AWS WAF a um estágio de API do API Gateway existente, use o comando `associate-web-acl` como no exemplo a seguir:

```
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 regional da web do AWS WAF a um estágio de API usando a API REST do AWS WAF

Para usar a API REST do AWS WAF visando associar uma ACL regional da web do AWS WAF a um estágio de API do API Gateway existente, use o comando `AssociateWebACL` como no exemplo a seguir:

```
import boto3

waf = boto3.client('wafregional')

waf.associate_web_acl(
    WebACLId='aabc123a-fb4f-4fc6-becb-2b00831cadcf',
    ResourceArn='arn:aws:apigateway:{region}::/restapis/4wk1k4onj3/stages/prod'
)
```

Criação e utilização de planos de uso com chaves de API

Depois que criar, testar e implantar suas APIs, você poderá usar planos de uso do API Gateway para disponibilizá-las como ofertas de produto para seus clientes. É possível configurar planos de uso e chaves de API para permitir que os clientes accessem APIs selecionadas com base em taxas de solicitação e cotas combinadas que atendam às suas necessidades de negócio e restrições orçamentárias. Se desejar, você pode definir as limitações de uso padrão em nível de método para uma API ou definir limitações para métodos de API específicos.

O que são planos de uso e chaves de API?

Um plano de uso especifica quem pode acessar um ou mais estágios e métodos da API implantada do —, além de quantos e em que velocidade eles podem ser acessados. O plano usa chaves de API para identificar clientes da API e calcula o acesso aos estágios de API associados para cada chave. Ele também permite que você configure limitações de uso e limites de cota que são aplicados a chaves de API de clientes individuais.

Chaves de API são valores de strings alfanuméricas distribuídas para desenvolvedores de aplicativos clientes para conceder acesso à sua API. Você pode usar chaves de API em conjunto com [planos de uso \(p. 450\)](#) ou [autorizadores do Lambda \(p. 396\)](#) para controlar o acesso às APIs. O API Gateway pode gerar chaves de API em seu nome, ou você pode importá-las de um [arquivo CSV \(p. 465\)](#). 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 usando o console do API Gateway” \(p. 454\)](#).

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 valor é uma string alfanumérica com tamanho entre 30 e 128 caracteres, por exemplo, `apikey1234abcdefghij0123456789`.

Important

Os valores de chaves de API devem ser exclusivos. Se você tentar criar duas chaves de API com diferentes nomes 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, determinada chave de API só pode ser associada a um único plano de uso para cada estágio de sua API.

Limitação de uso é um limite de taxa de solicitação aplicado a cada chave de API que você adiciona ao plano de uso. Você pode definir também uma limitação de uso padrão em nível de método para uma API ou definir limitações para métodos de API específicos.

Límite de cota é o número máximo de solicitações com determinada chave de API que podem ser enviadas em um intervalo de tempo especificado. Você pode configurar métodos de API individuais para exigir a autorização da chave de API com base na configuração do plano de uso. E você pode usar o comando [get-usage](#) da CLI e o método de API REST [usage : get](#) para determinar a quantidade de cota que um cliente da sua API usou.

Note

Controles de fluxo e limites de cota são aplicáveis a solicitações para chaves de API individuais que estão agregadas por todos os estágios de API de um plano de uso.

Melhores práticas para chaves de API e planos de uso

Veja a seguir sugestões das melhores práticas a serem seguidas ao usar chaves de API e planos de uso.

- Não dependa de chaves de API como o único meio de autenticação e autorização para as suas APIs. Por exemplo, 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 nesse plano de uso. Em vez disso, use uma função do IAM, [um autorizador do Lambda \(p. 396\)](#) ou um [grupo de usuários do Amazon Cognito \(p. 414\)](#).
- Se você estiver usando um [portal do desenvolvedor \(p. 664\)](#) para publicar suas APIs, observe que todas as APIs em um determinado plano de uso são podem ser assinadas, mesmo se você não as tornou visíveis para seus clientes.

Etapas para configurar um plano de uso

As etapas a seguir destacam como você, como proprietário da API, cria e configura um plano de uso para seus clientes.

Para configurar um plano de uso

1. Crie uma ou mais APIs, configure os métodos para exigir uma chave de API e implante as APIs em estágios.
2. Gere ou importe chaves de API para distribuir aos desenvolvedores de aplicativos (seus clientes) que usarão sua API.
3. Crie o plano de uso com os limites de controle de fluxo e cota desejados.
4. Associe estágios de API e chaves API ao plano de uso.

Os chamadores da API devem fornecer uma chave de API atribuída no cabeçalho `x-api-key` de solicitações à API.

Note

Para incluir métodos de API em um plano de uso, você deve configurar métodos de API individuais para [exigirem uma chave de API \(p. 454\)](#). Para autenticação e autorização de usuários, não use chaves de API. Use uma função do IAM, [um autorizador do Lambda \(p. 396\)](#) ou um [grupo de usuários do Amazon Cognito \(p. 414\)](#).

Escolha de uma chave de API

Ao associar um plano de uso a uma API e habilitar chaves de API em métodos de API, cada solicitação de entrada para a API deve conter uma [chave de API \(p. 6\)](#). O API Gateway lê a chave e a compara com as chaves no plano de uso. Se houver correspondência, o API Gateway regulará as solicitações de acordo com o limite de solicitação e a cota do plano. Caso contrário, ele emite uma exceção de `InvalidKeyParameter`. Como resultado, o chamador 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 do Amazon API Gateway” \(p. 408\)](#).

Para escolher uma origem de chave de API para uma API usando o console do API Gateway:

1. Faça login no console do API Gateway.
2. Escolha uma API existente ou crie uma nova.
3. No painel de navegação principal, escolha Settings (Configurações) sob a API escolhida ou recém-criada.
4. Na seção API Key Source (Origem da chave de API) do painel Settings (Configurações), escolha HEADER ou AUTHORIZER na lista suspensa.
5. Selecione Save Changes (Salvar alterações).

Para selecionar uma origem de chave de API para uma API usando a AWS CLI, chame o comando [update-rest-api](#) da seguinte forma:

```
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` para `HEADER` no comando da CLI acima.

Para selecionar uma origem de chave de API para uma API usando a API REST do API Gateway, chame [restapi:update](#) da seguinte forma:

```
PATCH /restapis/fugvwdxtri/ 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.

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.

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.

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 Lambda personalizado do tipo de token. Como uma propriedade no nível raiz da resposta de autorização, inclua `usageIdentifierKey:{api-key}`, onde `{api-key}` significa o valor da chave de API mencionado na etapa anterior.

4. Configure os métodos de API para exigir uma chave de API e ativar 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.

Configurar chaves de API usando o console do 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, você deve ter criado a API e tê-la implantado em uma etapa.

Para obter instruções sobre como criar e implantar uma API usando o console do API Gateway, consulte [Criação de uma API REST no Amazon API Gateway \(p. 182\)](#) e [Implantando uma API REST no Amazon API Gateway \(p. 516\)](#), respectivamente.

Tópicos

- [Exigir uma chave de API em um método \(p. 454\)](#)
- [Criar uma chave de API \(p. 455\)](#)
- [Importar as chaves de API \(p. 456\)](#)

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.

Para configurar um método de API para exigir uma chave de API

1. Faça login no Console de gerenciamento da AWS e abra o console do API Gateway em <https://console.aws.amazon.com/apigateway/>.
2. No painel de navegação principal do API Gateway, selecione Resources (Recursos).
3. Em Resources (Recursos), crie um novo método ou escolha um existente.
4. Escolha Method Request (Solicitação de método).
5. Na seção Authorization Settings (Configurações de autorização), escolha `true` para API Key Required (Chave de API necessária).
6. Selecione o ícone de marca de seleção para salvar as configurações.

Provide information about this method's authorization settings and the parameters it can receive.

Authorization Settings

Authorization: NONE ⓘ

API Key Required: true ⓘ

- ▶ URL Query String Parameters
- ▶ HTTP Request Headers
- ▶ Request Models [Create a Model](#)

7. Implante ou reimplemente a API para que o requisito entre em vigor.

Se a opção API Key Required (Chave de API necessária) 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.

Criar uma chave de API

Se você já tiver criado ou importado chaves de API para uso com planos de uso, poderá ignorar este procedimento e avançar para o seguinte.

Para criar uma chave de API

1. Faça login no Console de gerenciamento da AWS e abra o console do API Gateway em <https://console.aws.amazon.com/apigateway/>.
2. No painel de navegação principal do API Gateway, selecione API Keys (Chaves de API).
3. No menu suspenso Actions (Ações), selecione Create API key (Criar chave de API).

Select an API key

API Keys

Create API key ⓘ Import API keys

4. Em Create API Key (Criar chave de API), faça o seguinte:
 - a. Digite um nome de chave de API (por exemplo, **MyFirstKey**) no campo de entrada Name (Nome).

- b. Escolha Auto Generate (Gerar automaticamente) para que o API Gateway gere o valor da chave ou escolha Custom (Personalizado) para inserir a chave manualmente.
- c. Escolha Salvar.

Create API Key

Name*

API key* Auto Generate Custom

Description

* Required Save

5. Repita as etapas anteriores para criar mais chaves de API, se necessário.

Importar as chaves de API

O procedimento a seguir descreve como importar chaves de API para utilização com planos de uso.

Para importar chaves de API

1. No painel de navegação principal, selecione API Keys (Chaves de API).
2. No menu suspenso Actions (Ações), selecione Import API keys (Importar chaves de API).
3. Para carregar um arquivo de chave separado por vírgula, escolha Select CSV File (Selecionar arquivo CSV). Você também pode digitar as chaves manualmente. Para obter informações sobre o formato do arquivo, consulte [the section called "Formato de arquivo da chave de API do API Gateway" \(p. 465\)](#).

Import API Keys

Use the field below to upload your existing API Keys as comma separated values (CSV). API Keys will be created in API Gateway and associated with a Usage Plan. Learn about the CSV format in the [API Gateway documentation](#).

[Select CSV File](#)

1	name,Key,Description(enabled,usageplanIds
	2 ImportedKey,CWaiyZjNC212f9P7hcxG17Ae803jEdFu8pzfryqf,an imported key,true,abcdef

Fail on warnings Ignore warnings [Import](#)

4. Escolha Fail on warnings (Falha nos avisos) para interromper a importação em caso de erro ou escolha Ignore warnings (Ignorar avisos) para continuar a importar entradas de chave válidas quando houver um erro.
5. Para começar a importar as chaves de API selecionadas, escolha Import (Importar).

Agora que você configurou a chave de API, pode prosseguir para [criar e utilizar um plano de uso \(p. 457\)](#).

Criar, configurar e testar planos de uso com o console do API Gateway

Antes de criar um plano de uso, certifique-se de que você configurou as chaves de API desejadas. Para obter mais informações, consulte [Configurar chaves de API usando o console do API Gateway \(p. 454\)](#).

Esta seção descreve como criar e utilizar um plano de uso com o console do API Gateway.

Tópicos

- [Migrar a API para planos de uso padrão \(se necessário\) \(p. 457\)](#)
- [Criar um plano de uso \(p. 458\)](#)
- [Testar um plano de uso \(p. 461\)](#)
- [Manutenção de um plano de uso \(p. 461\)](#)

Migrar a API para planos de uso padrão (se necessário)

Se você começou a usar o API Gateway após o recurso de planos de uso ser lançado em 11 de agosto de 2016, você tem automaticamente os planos de uso habilitados em todas as regiões compatíveis.

Se começou a usar o API Gateway antes dessa data, pode ser necessário migrar para os planos de uso padrão. Você será solicitado com 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. No entanto, você deve usar a propriedade `UsagePlan apiStages` para associar valores de estágio de API especificados (`apiId` e `stage`) a chaves de API incluídas (via `UsagePlanKey`), em vez de usar a propriedade `ApiKey stageKeys`.

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` incluirá uma entrada de "UsagePlans" quando os planos de uso estiverem 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.

Para criar um plano de uso

1. No painel de navegação principal do Amazon API Gateway, escolha Usage Plans (Planos de uso) e depois Create (Criar).
2. Em Create Usage Plan (Criar plano de uso), faça o seguinte:
 - a. Em Name (Nome), digite um nome para o seu plano (por exemplo, `Plan_A`).
 - b. Em Description (Descrição), digite uma descrição para seu plano.
 - c. Selecione Enable throttling (Habilitar controle de utilização) e defina Rate (Taxa) (por exemplo, **100**) e Burst (Intermitência) (por exemplo, **200**).
 - d. Escolha Enable quota (Habilitar cota) e defina seu limite (por exemplo, **5000**) para um intervalo de tempo selecionado (por exemplo, Month (Mês)).
 - e. Escolha Next.

Create Usage Plan

Usage Plans help you meter API usage. With Usage Plans, you can enforce a throttling and quota limit on each API key. Throttling limits define the maximum number of requests per second available to each key. Quota limits define the number of requests each API key is allowed to make over a period.

Name* Plan_A

Description Sample usage plan

Throttling

Enable throttling

Rate* 100 requests per second i

Burst* 200 requests i

Quota

Enable quota

5000 requests per Month i

* Required Next

3. Para adicionar um estágio ao plano, faça o seguinte no painel Associated API Stages (Estágios de API associados):
 - a. Escolha Add API Stage (Adicionar estágio de API).
 - b. Escolha uma API (por exemplo, **PetStore**) na lista suspensa API.
 - c. Escolha um estágio (por exemplo, **Stage_1**) na lista suspensa Stage (Estágio).
 - d. Escolha o ícone de marca de seleção para salvar.

Associated API Stages

Add API Stage		
API	Stage	Method Throttling
PetStore	prod	No Methods Configured
Configure Method Throttling ×		

4. Para configurar o controle de utilização do método (p. 533), faça o seguinte:
 - a. Escolha Configure Method Throttling (Configurar controle de utilização do método).
 - b. Escolha Add Resource/Method (Adicionar recurso/método).
 - c. Escola o recurso no menu suspenso Resource (Recurso).

- d. Escolha o método no menu suspenso Method (Método).
 - e. Defina Rate (requests per second) (Taxa (solicitações por segundo)) (por exemplo, **100**) e Burst (Intermitência) (por exemplo, **200**).
 - f. Escolha o ícone de marca de seleção para salvar.
 - g. Escolha Fechar.
5. Para adicionar uma chave ao plano, faça o seguinte na guia API Keys (Chaves de API):
- a. Para usar uma chave existente, escolha Add API Key to Usage Plan (Adicionar chave de API ao plano de uso).
 - b. Para Name (Nome), digite um nome para a chave que você deseja adicionar (por exemplo, **MyFirstKey**).
 - c. Escolha o ícone de marca de seleção para salvar.
 - d. Conforme necessário, repita as etapas anteriores para adicionar outras chaves de API existentes a esse plano de uso.

Usage Plan API Keys

Subscribe an API key to this usage plan. Choose "Add API Key" below to search through your existing API keys. Once a key is associated with a plan, API Gateway will meter all requests from the key and apply the plan's throttling and quota limits.

The screenshot shows a user interface for managing API keys associated with a usage plan. At the top, there are two buttons: 'Add API Key to Usage Plan' and 'Create API Key and add to Usage Plan'. Below these is a dropdown menu set to 'Results per page 100'. A table lists an API key named 'MyFirstKey (Hiorr...)' with a checkbox next to it, which is checked. At the bottom, there are navigation arrows ('<', '>'), a page number 'Page 1', and two buttons: 'Back' and 'Done'.

Note

Alternativamente, para criar uma nova chave de API e adicioná-la ao plano de uso, escolha Create API Key and add to Usage Plan (Criar chave de API e adicionar ao plano de uso) e siga as instruções.

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, determinada chave de API só pode ser associada a um único plano de uso para cada estágio de sua API.

6. Para concluir a criação do plano de uso, escolha Done (Concluído).
7. Se quiser adicionar mais estágios de API ao plano de uso, escolha Add API Stage (Adicionar estágio da API) para repetir as etapas anteriores.

Testar um plano de uso

Para testar o plano de uso, você pode usar um SDK da AWS, a AWS CLI ou um cliente da API REST, como o Postman. Para ver um exemplo de como usar o [Postman](#) para testar o plano de uso, consulte [Testar planos de uso \(p. 464\)](#).

Manutenção de um plano de uso

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 e estender quotas.

Para monitorar as cotas usadas e restantes

1. No painel de navegação principal do API Gateway, escolha Usage Plans (Planos de uso).
2. Escolha um plano de uso na lista de planos de uso.
3. No plano especificado, escolha API Keys (Chaves de API).
4. Escolha uma chave de API e, em seguida, escolha Usage (Uso) para visualizar Subscriber's Traffic (Tráfego do assinante) no plano que você está monitorando.
5. Opcionalmente, escolha Export (Exportar), selecione uma data From (De) e uma data To (Até), escolha **JSON** ou **CSV** para o formato de dados exportado e depois selecione Export (Exportar).

O exemplo a seguir mostra um arquivo exportado.

```
{  
    "thisPeriod": {  
        "px1KW6...qBazOJH": [  
            [  
                0,  
                5000  
            ],  
            [  
                0,  
                5000  
            ],  
            [  
                0,  
                10  
            ]  
        ],  
        "startDate": "2016-08-01",  
        "endDate": "2016-08-03"  
    }  
}
```

Os dados de uso no exemplo mostram os dados de uso diários para um cliente de API, conforme identificado pela chave de API (`px1KW6...qBazOJH`), 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.

Para estender as cotas restantes

1. Repita as etapas de 1 – 3 do procedimento anterior.
2. No painel do plano de uso, escolha Extension (Extensão) na janela do plano de uso.
3. Digite um número para as cotas de solicitação Remaining (Restante).
4. Escolha Salvar.

Configurar chaves de API usando a API REST do 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, você deve ter criado a API e tê-la implantado em uma etapa.

Para as chamadas da API REST a fim de criar e implantar uma API, consulte [restapi:create](#) e [deployment:create](#), respectivamente.

Tópicos

- [Exigir uma chave de API em um método \(p. 462\)](#)
- [Criar ou importar chaves de API \(p. 462\)](#)

Exigir uma chave de API em um método

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 ou importar chaves de API

Para criar ou importar uma chave de API, siga um destes procedimentos:

- Chame `apikey:create` para criar uma chave de API.
- Chame `apikey:import` para importar uma chave de API de um arquivo. Para conhecer o formato de arquivo, consulte [Formato de arquivo da chave de API do API Gateway \(p. 465\)](#).

Com a chave de API criada, você pode agora prosseguir para [Criar, configurar e testar planos de uso com a CLI do API Gateway e API REST \(p. 462\)](#).

Criar, configurar e testar planos de uso com a CLI do API Gateway e API REST

Antes de configurar um plano de uso, você já deve ter feito o seguinte: configurado métodos de uma API selecionada para exigir chaves de API, implantado ou reimplantado a API em um estágio e criado ou importado uma ou mais chaves de API. Para obter mais informações, consulte [Configurar chaves de API usando a API REST do API Gateway \(p. 462\)](#).

Para configurar um plano de uso usando a API REST do API Gateway, use as seguintes instruções, supondo que você já criou as APIs a serem adicionadas ao plano de uso.

Tópicos

- [Migrar para planos de uso padrão \(p. 463\)](#)
- [Criar um plano de uso \(p. 463\)](#)
- [Gerenciar um plano de uso com a CLI da AWS \(p. 464\)](#)
- [Testar planos de uso \(p. 464\)](#)

Migrar para planos de uso padrão

Ao criar um plano de uso pela primeira vez, é possível migrar estágios de API existentes que estão associadas a chaves de API selecionadas para um plano de uso, chamando `account:update` com o seguinte corpo:

```
{  
    "patchOperations" : [ {  
        "op" : "add",  
        "path" : "/features",  
        "value" : "UsagePlans"  
    } ]  
}
```

Para obter mais informações sobre como migrar estágios da API associados às chaves de API, consulte [Migrar para planos de uso padrão no console do API Gateway \(p. 457\)](#).

Criar um plano de uso

O procedimento a seguir descreve como criar um plano de uso.

Para criar um plano de uso com a API REST

1. Chame `usageplan:create` para criar um plano de uso. Na carga, especifique o nome e a descrição do plano, estágios de API associados, limites de taxa e cotas.

Anote o identificador de plano de uso resultante. Você precisa dele na próxima etapa.

2. Siga um destes procedimentos:

- a. Chame `usageplankey:create` para adicionar uma chave de API ao plano de uso. Especifique `keyId` e `keyType` na carga.

Para adicionar mais chaves de API ao plano de uso, repita a chamada anterior, trabalhando com uma chave de API de cada vez.

- b. 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 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.

Gerenciar um plano de uso com a CLI da AWS

Os exemplos de código a seguir mostram como adicionar, remover ou modificar as configurações de limitação em nível de método em um plano de uso chamando o comando `update-usage-plan`.

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='"\\"/pets/GET\\":{\\\"rateLimit\\":1.0,\\\"burstLimit\\\":1},\\\"//GET\\":{\\\"rateLimit\\":1.0,\\\"burstLimit\\\":1}\\}"'
```

Testar planos de uso

Como exemplo, vamos usar a API PetStore, que foi criada em [TUTORIAL: Crie uma API REST importando um exemplo \(p. 45\)](#). 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, xbvxlpjch) 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: xbvxlpjch.execute-api.ap-southeast-1.amazonaws.com
X-Amz-Date: 20160803T001845Z
Authorization: AWS4-HMAC-SHA256 Credential={access_key_ID}/20160803/ap-southeast-1/
execute-api/aws4_request, SignedHeaders=content-type;host;x-amz-date;x-api-key,
Signature={sigv4_hash}
```

Note

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` requerido.

A resposta bem-sucedida retorna um código de status 200 `OK` e uma carga que contém os resultados solicitados do back-end. 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 atende a essa solicitação sem aplicar esses limites ou cotas, conforme especificado no plano de uso. Mas registra uma mensagem de erro de `Usage Plan check failed due to an internal error` no CloudWatch. Você pode ignorar esses erros ocasionais.

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
apikey1234abcdefg hij0123456789,MyFirstApiKey
```

Um valor de `Key` deve ter entre 30 e 128 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,glvrsr"
```

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.

Documentação de uma API REST no API Gateway

Para ajudar os clientes a compreender e usar sua API, você deve documentá-la. Para ajudá-lo a documentar sua API, o API Gateway permite adicionar e atualizar o conteúdo da ajuda para entidades da 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 sua API, chame a [API REST do API Gateway](#), use um dos [SDKs da AWS](#) ou [AWS CLIs](#) do API Gateway, ou use o console do API Gateway. Além disso, é possível importar ou exportar as partes da documentação definidas em um arquivo do OpenAPI externo. Antes de explicarmos como documentar sua API, mostraremos como a documentação da API é representada no API Gateway.

Tópicos

- [Representação da documentação da API no API Gateway \(p. 466\)](#)
- [Documentar uma API usando o console do API Gateway \(p. 474\)](#)
- [Publicar a documentação da API usando o console do API Gateway \(p. 483\)](#)
- [Documentar uma API usando a API REST do API Gateway \(p. 483\)](#)
- [Publicar a documentação da API usando a API REST do API Gateway \(p. 498\)](#)
- [Importar a documentação da API \(p. 505\)](#)
- [Controlar o acesso à documentação da API \(p. 509\)](#)

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 \(p. 466\)](#)
- [Versões da documentação \(p. 474\)](#)

Partes da documentação

Um recurso [DocumentationPart](#) é um objeto JSON que armazena o conteúdo da documentação aplicável a uma entidade da API individual. 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 \(p. 468\)](#).

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 da API pode ser um recurso da 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 de 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 do OpenAPI. O API Gateway propaga atributos de conteúdo compatíveis com o OpenAPI para as definições de entidades da API das instâncias `DocumentationPart` associadas. O API Gateway não propaga os atributos de conteúdo não compatíveis nas definições de entidades da 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` para uma instância `DocumentationPart` com esse tipo de padrão `location`. O API Gateway extrai os atributos da documentação do OpenAPI de `DocumentationPart` da entidade da API genérica e os injeta em uma entidade da 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 `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	<code>path</code>	<code>statusCode</code>	<code>name</code>	Observação
1	/pets	*	id	A documentação

Caso	path	statusCode	name	Observações
				<p>associada a esse padrão de localização será herdada por entidades que corresponderem ao padrão de localização.</p>
2	/pets	200	id	<p>A documentação associada a esse padrão de localização será herdada por entidades que corresponderem ao padrão de localização quando tanto o Caso 1 quanto o Caso 2 forem correspondidos, pois o Caso 2 é mais específico do que o Caso 1.</p>

Caso	path	statusCode	name	Observações
3	/pets/ petId	*	id	A documentação associada a esse padrão de localização será herdada por entidades que corresponderem ao padrão de localização quando os Casos 1, 2 e 3 forem correspondidos, pois o Caso 3 tem precedência maior do que o Caso 2 e é mais específico do que o Caso 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
API	{ "location": { "type": "API" }, ... }	type	N/D	Não
Recurso	{ "location": { "type": "RESOURCE", "path": " resource_path " }, ... }	type	O valor padrão de path é /.	Não
Método	{ "location": { "type": "METHOD", "path": " resource_path ", "method": " http_verb " }, ... }	type	Os valores padrão de path e method são / e *, respectivamente.	Sim, correspondendo path por prefixo e correspondendo method de quaisquer valores.
Parâmetro de consulta	{ "location": { "type": " QUERY_PARAMETER ", "path": " resource_path ", "method": " HTTP_verb ", }	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
	<pre> "name": "query_parameter_name" }, ... } </pre>			
Corpo da solicitação	<pre> { "location": { "type": "REQUEST_BODY", "path": "resource_path", "method": "http_verb" }, ... } </pre>	type	Os valores padrão de path e method são / e *, respectivamente.	Sim, correspondendo path por prefixo e correspondendo method por valores exatos.
Parâmetro do cabeçalho da solicitação	<pre> { "location": { "type": "REQUEST_HEADER", "path": "resource_path", "method": "HTTP_verb", "name": "header_name" }, ... } </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": "resource/ {path_parameter_name}", "method": "HTTP_verb", "name": "path_parameter_name" }, ... } </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": "resource_path", "method": "http_verb", "statusCode": "status_code" }, ... }</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": "resource_path", "method": "http_verb", "statusCode": "status_code", "name": "header_name" }, ... }</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.
Corpo da resposta	<pre>{ "location": { "type": "RESPONSE_BODY", "path": "resource_path", "method": "http_verb", "statusCode": "status_code" }, ... }</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.
Autorizador	<pre>{ "location": { "type": "AUTHORIZER", "name": "authorizer_name" }, ... }</pre>	type	N/D	Não

Entidade de API	Campos de localização válidos	Campos de localização obrigatórios	Valores de campo padrão	Conteúdo herdável
Modelo	<pre>{ "location": { "type": "MODEL", "name": "model_name" }, ... }</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.

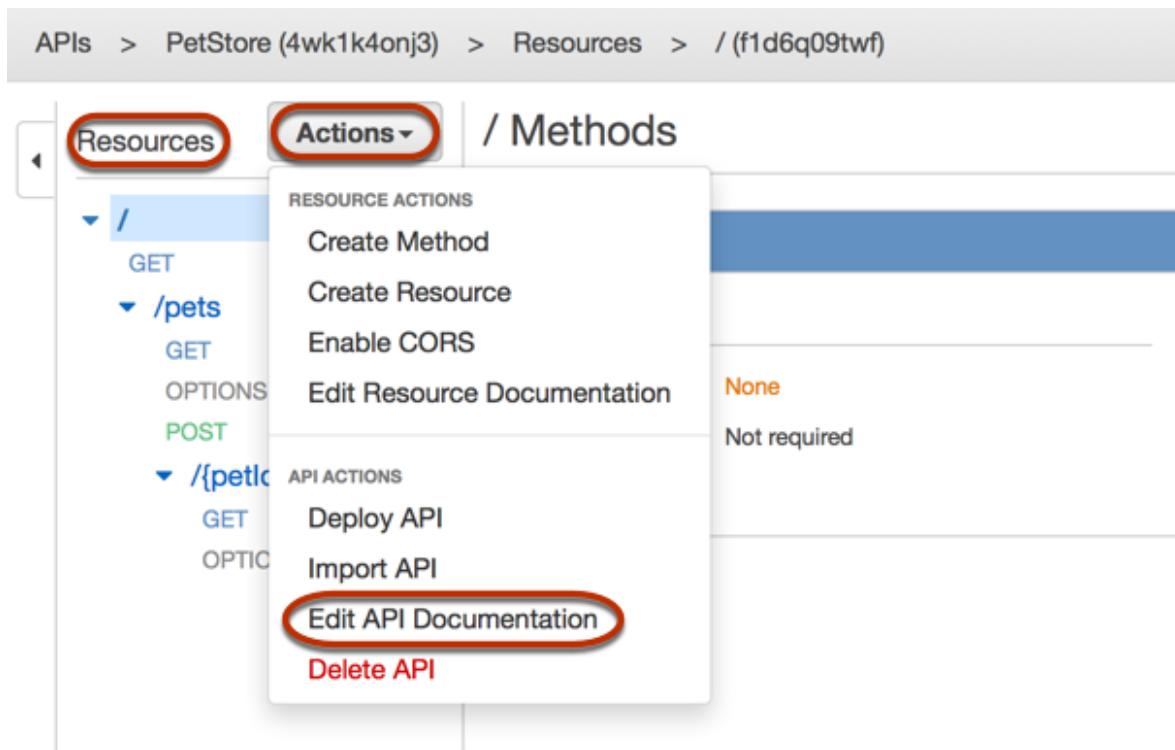
Um pré-requisito para criar e editar a documentação de uma API é que você já deve ter criado essa API. Nesta seção, usamos a API [PetStore](#) como exemplo. Para criar uma API usando o console do API Gateway, siga as instruções em [TUTORIAL: Crie uma API REST importando um exemplo \(p. 45\)](#).

Tópicos

- [Documentar a entidade API \(p. 474\)](#)
- [Documentar uma entidade RESOURCE \(p. 477\)](#)
- [Documentar uma entidade METHOD \(p. 478\)](#)
- [Documentar uma entidade QUERY_PARAMETER \(p. 478\)](#)
- [Documentar uma entidade PATH_PARAMETER \(p. 479\)](#)
- [Documentar uma entidade REQUEST_HEADER \(p. 480\)](#)
- [Documentar uma entidade REQUEST_BODY \(p. 480\)](#)
- [Documentar uma entidade RESPONSE \(p. 480\)](#)
- [Documentar uma entidade RESPONSE_HEADER \(p. 481\)](#)
- [Documentar uma entidade RESPONSE_BODY \(p. 481\)](#)
- [Documentar uma entidade MODEL \(p. 481\)](#)
- [Documentar uma entidade AUTHORIZER \(p. 482\)](#)

Documentar a entidade API

Para adicionar uma parte de documentação para a entidade API, escolha Resources (Recursos) na API PetStore. Selecione o item de menu Actions → Edit API Documentation (Ações → Editar API).



Se uma parte de documentação não tiver criada para a API, você acessará o editor de mapa de properties da parte da documentação. Digite o seguinte mapa de properties no editor de texto e selecione Save (Salvar) para criar a parte da documentação.

```
{  
  "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.

Documentation

Provide your API documentation in JSON format in the form below.

Type API

```
1  {
2      "info": {
3          "description" : "Your first API Gateway API.",
4          "contact": {
5              "name": "John Doe",
6              "email": "john.doe@api.com"
7          }
8      }
9 }
```

Close **Save**

Se uma parte de documentação já tiver sido criada, primeiro você obterá o visualizador de mapa de **properties**, conforme mostrado a seguir.

Documentation

Specify your documentation part in JSON format in the form below. For more information, see the [Documentation Parts Documentation](#).

Type API

```
{  
  "info": {  
    "description": "Your first API Gateway API.",  
    "contact": {  
      "name": "John Doe",  
      "email": "john.doe@api.com"  
    }  
  }  
}
```

Delete **Close** **Edit**

Escolher Edit (Editar) ativa o editor de mapa de `properties`, conforme mostrado anteriormente.

Documentar uma entidade RESOURCE

Para adicionar ou editar a parte de documentação do recurso raiz da API, escolha / na árvore Resource (Recurso) e depois escolha o item de menu Actions → Edit Resource Documentation (Ações → Editar a documentação do recurso).

Se nenhuma parte de documentação tiver sido criada para essa entidade, você verá a janela Documentation (Documentação). Digite um mapa de `properties` válido no editor. Em seguida, escolha Save (Salvar) e Close (Fechar).

```
{  
  "description": "The PetStore's root resource."  
}
```

Se uma parte de documentação já tiver definida para a entidade RESOURCE, você obterá o visualizador de documentação. Escolha Edit (Editar) para abrir o editor de Documentation. Modifique o mapa de `properties` existente. Escolha Save (Salvar) e escolha Close (Fechar).

Se necessário, repita essas etapas para adicionar uma parte de documentação a outras entidades RESOURCE.

Documentar uma entidade METHOD

Para adicionar ou editar a documentação de uma entidade METHOD, usando o método GET no recurso raiz como exemplo, escolha GET no recurso / e depois escolha o item de menu Actions → Edit Method Documentation (Ações → Editar a documentação do método).

Para a nova parte da documentação, digite o seguinte mapa de properties no editor de Documentation (Documentação) na janela Documentation (Documentação). Em seguida, escolha Save (Salvar) e Close (Fechar).

```
{  
  "tags" : [ "pets" ],  
  "description" : "PetStore HTML web page containing API usage information"  
}
```

Para a documentação existente, escolha Edit (Editar) no visualizador Documentation (Documentação). Edite o conteúdo da documentação no editor Documentation (Documentação) e escolha Save (Salvar). Escolha Fechar.

No visualizador Documentation (Documentação), você também pode excluir a parte da documentação.

Se necessário, repita essas etapas para adicionar uma parte de documentação a outros métodos.

Documentar uma entidade QUERY_PARAMETER

Para adicionar ou editar uma parte de documentação de um parâmetro de consulta de solicitação, usando o método GET /pets?type=...&page=... como exemplo, escolha GET em /pets, na árvore Resources (Recursos). Escolha Method Request (Solicitação de método) na janela Method Execution (Execução de método). Expanda a seção URL Query String Parameters (Parâmetros da string de consulta de URL). Escolha o parâmetro de consulta page (página), por exemplo, e escolha o ícone de livro para abrir o visualizador ou o editor Documentation (Documentação).

The screenshot shows the AWS API Gateway Method Execution configuration interface. At the top, it displays the path `/pets - GET - Method Request`. Below this, there's a section for Authorization Settings, which shows "Authorization NONE" and "API Key Required true". A large red oval highlights the "URL Query String Parameters" section. Under this section, the "page" parameter is listed with its details: "Name: page", "Caching: off", and two small icons for edit and cancel. There are also "Add query string" and "Delete" buttons. The entire screenshot is framed by a thin gray border.

Como alternativa, é possível escolher Documentation (Documentação) sob a API PetStore, no painel de navegação principal. Então, escolha **Query Parameter** para Type (Tipo). Para a API PetStore de exemplo, isso mostra as partes de documentação dos parâmetros de consulta `page` e `type`.

The screenshot shows the 'Documentation' section of the AWS Lambda console. At the top, there are buttons for 'Create Documentation Part', 'Import Documentation', and 'Publish Documentation'. Below this, a text area provides instructions for adding documentation to help developers understand how to interact with the API. It mentions that documentation parts can be shared across multiple resources and methods by specifying a wildcard value (*) for method or status code, and that documentation for a 200 response can be used in multiple locations. It also notes that documentation can be imported by supplying a Swagger definition file and published to a stage. For more information, it references the [documentation](#).

Below the instructions, there are two side-by-side panels for defining query parameters:

- Left Panel (page):** Type is set to 'Query Parameter'. Path is '/pets'. Method is 'GET'. Name is 'page'. Description: '{ "description": "Page number of results to return." }'
- Right Panel (type):** Type is set to 'Query Parameter'. Path is '/pets'. Method is 'GET'. Name is 'type'. Description: '{ "description": "The type of pet to retrieve" }'

Both panels have 'Edit' and 'Clone' buttons at the bottom.

Para uma API com parâmetros de consulta definidos para outros métodos, você pode filtrar sua seleção, especificando o path do recurso afetado para Path (Caminho), escolhendo o método HTTP desejado em Method (Método), ou digitando o nome do parâmetro de consulta em Name (Nome).

Por exemplo, escolha o parâmetro de consulta `page`. Escolha Edit (Editar) para modificar a documentação existente. Escolha Save (Salvar) para salvar a alteração.

Para adicionar uma nova parte de documentação para uma entidade `QUERY_PARAMETER`, escolha Create Documentation Part (Criar parte da documentação). Escolha `Query Parameter` para Type (Tipo). Digite um caminho de recurso (por exemplo, `/pets`) em Path (Caminho). Escolha um verbo HTTP (por exemplo, `GET`) para Method (Método). Digite uma descrição de properties no editor de texto. Em seguida, escolha Save (Salvar).

Se necessário, repita essas etapas para adicionar uma parte de documentação a outros parâmetros de consulta de solicitação.

Documentar uma entidade PATH_PARAMETER

Para adicionar ou editar a documentação de um parâmetro de caminho, vá para Method Request (Solicitação de método) do método no recurso especificado pelo parâmetro de caminho. Expanda a seção Request Paths (Caminhos de solicitação). Escolha o ícone de livro do parâmetro de caminho para abrir o visualizador ou editor Documentation (Documentação). Adicione ou modifique as propriedades da parte da documentação.

Como alternativa, escolha Documentation (Documentação) sob a API PetStore, no painel de navegação principal. Escolha `Path Parameter` para Type (Tipo). Escolha Edit (Editar) em um parâmetro de caminho na lista. Modifique o conteúdo e escolha Save (Salvar).

Para adicionar a documentação de um parâmetro de caminho não listado, escolha Create Documentation Part (Criar parte da documentação). Escolha `Path Parameter` (Parâmetro do caminho) para Type (Tipo). Defina um caminho de recurso em Path (Caminho), escolha um método em Method (Método) e defina um nome de parâmetro de caminho em Name (Nome). Adicione as propriedades da documentação e escolha Save (Salvar).

Se necessário, repita essas etapas para editar ou adicionar uma parte de documentação a outros parâmetros de caminho.

Documentar uma entidade REQUEST_HEADER

Para adicionar ou editar a documentação de um cabeçalho de solicitação, vá para o Method Request (Solicitação de método) do método com o parâmetro de cabeçalho. Expanda a seção HTTP Request Headers (Cabeçalhos da solicitação HTTP). Escolha o ícone de livro do cabeçalho para abrir o visualizador ou editor Documentation (Documentação). Adicione ou modifique as propriedades da parte da documentação.

Como alternativa, escolha Documentation (Documentação) sob a API, no painel de navegação principal. Então, escolha Request Header para Type (Tipo). Escolha Edit (Editar) em um cabeçalho de solicitação listado para alterar a documentação. Para adicionar a documentação de um cabeçalho de solicitação não listado, escolha Create Documentation Part (Criar parte da documentação). Escolha Request Header (Cabeçalho da solicitação) para Type (Tipo). Especifique um caminho de recurso em Path (Caminho). Escolha um método para Method (Método). Digite um nome de cabeçalho em Name (Nome). Em seguida, adicione e salve um mapa de properties.

Se necessário, repita essas etapas para editar ou adicionar uma parte de documentação a outros cabeçalhos de solicitação.

Documentar uma entidade REQUEST_BODY

Para adicionar ou editar a documentação de um corpo de solicitação, acesse Method Request (Solicitação de método) de um método. Escolha o ícone de livro para Request Body (Corpo da solicitação), para abrir o visualizador Documentation (Documentação) e depois o editor. Adicione ou modifique as propriedades da parte da documentação.

Como alternativa, escolha Documentation (Documentação) sob a API, no painel de navegação principal. Então, escolha Request Body para Type (Tipo). Escolha Edit (Editar) em um corpo de solicitação listado para alterar a documentação. Para adicionar a documentação de um corpo de solicitação não listado, escolha Create Documentation Part (Criar parte da documentação). Escolha Request Body para Type (Tipo). Defina um caminho de recurso em Path (Caminho). Escolha um verbo HTTP para Method (Método). Em seguida, adicione e salve um mapa de properties.

Se necessário, repita essas etapas para editar ou adicionar uma parte de documentação a outros corpos de solicitação.

Documentar uma entidade RESPONSE

Para adicionar ou editar a documentação de uma resposta, acesse Method Response (Resposta de método) de um método. Selecione o ícone de livro para Method Response (Resposta de método), para abrir o visualizador da Documentation (Documentação) e depois o editor. Adicione ou modifique as propriedades da parte da documentação.

Provide information about this method's response types, their headers and content types.

HTTP Status
200

[+ Add Response](#)

Como alternativa, escolha Documentation (Documentação) sob a API, no painel de navegação principal. Então, escolha Response (status code) para Type (Tipo). Escolha Edit (Editar) em uma resposta listada de um código de status HTTP especificado para alterar a documentação. Para adicionar a

documentação de um corpo de resposta não listado, escolha Create Documentation Part (Criar parte da documentação). Escolha Response (status code) (Resposta (código de status)) para Type (Tipo). Defina um caminho de recurso em Path (Caminho). Escolha um verbo HTTP para Method (Método). Digite um código de status HTTP em Status Code (Código de status). Em seguida, adicione e salve as propriedades da parte da documentação.

Se necessário, repita essas etapas para editar ou adicionar uma parte de documentação a outras respostas.

Documentar uma entidade RESPONSE_HEADER

Para adicionar ou editar a documentação de um cabeçalho de resposta, acesse Method Response (Resposta de método) de um método. Expanda uma seção de resposta de um determinado status HTTP. Escolha o ícone de livro de um cabeçalho de resposta em Response Headers for **Status Code** (Cabeçalhos de resposta para StatusCode) para abrir o visualizador Documentation (Documentação) e depois o editor. Adicione ou modifique as propriedades da parte da documentação.

Como alternativa, escolha Documentation (Documentação) sob a API, no painel de navegação principal. Então, escolha Response Header para Type (Tipo). Escolha Edit (Editar) em um cabeçalho de resposta listado para alterar a documentação. Para adicionar a documentação de um cabeçalho de resposta não listado, escolha Create Documentation Part (Criar parte da documentação). Escolha Response Header (Cabeçalho da resposta) para Type (Tipo). Defina um caminho de recurso em Path (Caminho). Escolha um verbo HTTP para Method (Método). Digite um código de status HTTP em Status Code (Código de status). Digite o nome do cabeçalho de resposta em Name (Nome). Em seguida, adicione e salve as propriedades da parte da documentação.

Se necessário, repita essas etapas para editar ou adicionar uma parte de documentação a outros cabeçalhos de resposta.

Documentar uma entidade RESPONSE_BODY

Para adicionar ou editar a documentação de um corpo de resposta, acesse Method Response (Resposta do método) de um método. Expanda a seção de resposta de um determinado status HTTP. Escolha o ícone de livro para Response Body for **Status Code** (Corpo da resposta para StatusCode) para abrir o visualizador Documentation (Documentação) e depois o editor. Adicione ou modifique as propriedades da parte da documentação.

Como alternativa, escolha Documentation (Documentação) sob a API, no painel de navegação principal. Então, escolha Response Body para Type (Tipo). Escolha Edit (Editar) em um corpo de resposta listado para alterar a documentação. Para adicionar a documentação de um corpo de resposta não listado, escolha Create Documentation Part (Criar parte da documentação). Escolha Response Body (Corpo da resposta) para Type (Tipo). Defina um caminho de recurso em Path (Caminho). Escolha um verbo HTTP para Method (Método). Digite um código de status HTTP em Status Code (Código de status). Em seguida, adicione e salve as propriedades da parte da documentação.

Se necessário, repita essas etapas para editar ou adicionar uma parte de documentação a outros corpos de resposta.

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 ou editar documentação de um modelo, acesse `Models` (Modelos) da API, no painel de navegação principal. Escolha o ícone de livro para o nome de um modelo listado para abrir o visualizador `Documentation` (Documentação) e depois o editor. Adicione ou modifique as propriedades da parte da documentação.

Como alternativa, escolha `Documentation` (Documentação) sob a API, no painel de navegação principal. Então, escolha `Model` para `Type` (Tipo). Escolha `Edit` (Editar) em um modelo listado para alterar a documentação. Para adicionar a documentação de um modelo não listado, escolha `Create Documentation Part` (Criar parte da documentação). Escolha o `Model` (Modelo) para `Type` (Tipo). Dê um nome ao modelo em `Name` (Nome). Em seguida, adicione e salve as propriedades da parte da documentação.

Se necessário, repita essas etapas para editar ou adicionar uma parte de documentação a outros modelos.

Documentar uma entidade `AUTHORIZER`

Para adicionar ou editar a documentação de um autorizador, acesse `Authorizers` (Autorizadores) da API, no painel de navegação principal. Escolha o ícone de livro do autorizador listado para abrir o visualizador `Documentation` (Documentação) e depois o editor. Adicione ou modifique as propriedades da parte da documentação.

Como alternativa, escolha `Documentation` (Documentação) sob a API, no painel de navegação principal. Então, escolha `Authorizer` para `Type` (Tipo). Escolha `Edit` (Editar) em um autorizador listado para alterar a documentação. Para adicionar a documentação de um autorizador não listado, escolha `Create Documentation Part` (Criar parte da documentação). Escolha `Authorizer` (Autorizador) para `Type` (Tipo). Dê um nome ao autorizador em `Name` (Nome). Em seguida, adicione e salve as propriedades da parte da documentação.

Se necessário, repita essas etapas para editar ou adicionar uma parte de documentação a outros autorizadores.

Para adicionar uma parte de documentação para um autorizador, escolha Create Documentation Part (Criar parte da documentação). Escolha Authorizer (Autorizador) para Type (Tipo). Especifique um valor para o campo válido location de Name (Nome) para o autorizador.

Adicione e salve o conteúdo da documentação no editor de mapas de properties.

Se necessário, repita essas etapas para adicionar uma parte de documentação a outro autorizador.

Publicar a documentação da API usando o console do API Gateway

O procedimento a seguir descreve como publicar uma versão de documentação.

Para publicar uma versão de documentação usando o console do API Gateway

1. Escolha Documentation (Documentação) para a API no painel de navegação principal do console API Gateway.
2. Escolha Publish Documentation (Publicar documentação) no painel Documentation (Documentação).
3. Configure a publicação:
 - a. Escolha um nome disponível para Stage (Estágio).
 - b. Digite um identificador de versão, por exemplo, 1.0.0, em Version (Versão).
 - c. Opcionalmente, forneça uma descrição sobre a publicação em Description (Descrição).
4. Escolha Publish.

Agora, você pode prosseguir e fazer download da documentação publicada, exportando a documentação para um arquivo do OpenAPI externo.

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: Crie uma API REST importando um exemplo \(p. 45\)](#).

Tópicos

- [Documentar a entidade API \(p. 484\)](#)
- [Documentar uma entidade RESOURCE \(p. 485\)](#)
- [Documentar uma entidade METHOD \(p. 487\)](#)
- [Documentar uma entidade QUERY_PARAMETER \(p. 490\)](#)
- [Documentar uma entidade PATH_PARAMETER \(p. 491\)](#)
- [Documentar uma entidade REQUEST_BODY \(p. 491\)](#)
- [Documentar uma entidade REQUEST_HEADER \(p. 492\)](#)
- [Documentar uma entidade RESPONSE \(p. 493\)](#)

- Documentar uma entidade RESPONSE_HEADER (p. 494)
- Documentar uma entidade AUTHORIZER (p. 495)
- Documentar uma entidade MODEL (p. 496)
- Atualizar partes da documentação (p. 498)
- Listar partes de documentação (p. 498)

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: 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" : "API"
    },
    "properties": "{\n\t\"info\": {\n\t\t\"description\" : \"Your first API with Amazon API\nGateway.\n\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\nGateway.\n\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: 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\"info\": {\n\t\t\"description\" : \"Your first API with Amazon API\nGateway.\n\t}\n}\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: 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" : "RESOURCE",
  },
  "properties" : "{\n\t\"description\" : \"The PetStore root resource.\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/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: 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" : "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: 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" : "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/o64bjbj"
        },
        "documentationpart:delete": {
            "href": "/restapis/4wk1k4onj3/documentation/parts/o64bjbj"
        },
        "documentationpart:update": {
            "href": "/restapis/4wk1k4onj3/documentation/parts/o64bjbj"
        }
    },
    "id": "o64bjbj",
    "location": {
        "path": "/pets",
        "method": "GET",
        "name": null,
        "statusCode": null,
        "type": "METHOD"
    },
    "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/o64bjbj"
        },
        "documentationpart:delete": {
            "href": "/restapis/4wk1k4onj3/documentation/parts/o64bjbj"
        },
        "documentationpart:update": {
            "href": "/restapis/4wk1k4onj3/documentation/parts/o64bjbj"
        }
    },
    "id": "o64bjbj",
    "location": {
```

```
    "path": "/pets",
    "method": "GET",
    "name": null,
    "statusCode": null,
    "type": "METHOD"
},
"properties": "{\n\t\"summary\" : \"List all pets.\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}"
  } ]
}
```

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/o64bjbj"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64bjbj"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/o64bjbj"
    }
  },
  "id": "o64bjbj",
  "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\"}\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/partsh9ht5w"
        },
        "documentationpart:delete": {
            "href": "/restapis/4wk1k4onj3/documentation/partsh9ht5w"
        },
        "documentationpart:update": {
            "href": "/restapis/4wk1k4onj3/documentation/partsh9ht5w"
        }
    },
    "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\"}\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}"
}
```

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\t\"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: 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_HEADER",
        "path" : "/pets",
        "method" : "GET",
        "name" : "x-my-token"
    },
    "properties": "{\n\t\"description\" : \"A custom token used to authorization the method\ninvocation.\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/partsh0m3uf"
        },
        "documentationpart:delete": {
            "href": "/restapis/4wk1k4onj3/documentation/partsh0m3uf"
        },
        "documentationpart:update": {
            "href": "/restapis/4wk1k4onj3/documentation/partsh0m3uf"
        }
    },
    "id": "h0m3uf",
    "location": {
        "path": "/pets",
        "method": "GET",
        "name": "x-my-token",
        "statusCode": null,
        "type": "REQUEST_HEADER"
    },
    "properties": "{\n\t\"description\" : \"A custom token used to authorization the method\ninvocation.\n\"}\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/partsh0m3uf 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:

```
{\n  "_links": {\n    "self": {\n      "href": "/restapis/4wk1k4onj3/documentation/parts/lattew"\n    },\n    "documentationpart:delete": {\n      "href": "/restapis/4wk1k4onj3/documentation/parts/lattew"\n    },\n    "documentationpart:update": {\n      "href": "/restapis/4wk1k4onj3/documentation/parts/lattew"\n    }\n  },\n  "id": "lattew",\n  "location": {\n    "path": "/",\n    "method": "*",\n    "name": null,\n    "statusCode": "200",\n    "type": "RESPONSE"\n  },\n  "properties": "{\n    \"description\" : \"Successful operation.\\"\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\nHost: apigateway.region.amazonaws.com\nContent-Type: application/json\nX-Amz-Date: YYYYMMDDTTtttttZ\nAuthorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/  
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,  
Signature=sigv4_secret\n\n"location": {\n  "path": "/",
  "method": "GET",
  "name": "Content-Type",
  "statusCode": "200",
  "type": "RESPONSE_HEADER"
},\n"properties": "{\n  \"description\" : \"Media type of request\"\n}"
```

Se bem-sucedida, a operação retornará uma resposta 201 `Created`, contendo a instância `DocumentationPart` recém-criada na carga útil. por exemplo:

```
{\n  "_links": {\n    "curies": {\n      "href": "http://docs.aws.amazon.com/apigateway/latest/developerguide/restapi-\n      documentationpart-{rel}.html",\n      "name": "documentationpart",\n      "templated": true\n    }
  }
}
```

```
},
"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    \"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 DocumentationPart 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: YYYYMMDDTtttttz
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
  "location" : {
    "type" : "MODEL",
    "name" : "Pet"
  },
  "properties": "{\n\t\"description\" : \"Data structure of a Pet object.\n\"}"
}
```

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/lkn4uq"
    },
    "documentationpart:delete": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/lkn4uq"
    },
    "documentationpart:update": {
      "href": "/restapis/4wk1k4onj3/documentation/parts/lkn4uq"
    }
  },
  "id": "lkn4uq",
  "location": {
    "path": null,
    "method": null,
    "name": "Pet",
    "statusCode": null,
    "type": "MODEL"
  },
  "properties": "{\n\t\"description\" : \"Data structure of a Pet object.\n\"}"
}
```

Repita 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: YYYYMMDDTTtttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
  "patchOperations" : [ {
    "op" : "replace",
    "path" : "RESOURCE_PATH",
    "value" : "NEW_properties_VALUE_AS_JSON_STRING"
  } ]
}
```

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 de 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 \(p. 499\)](#)
- [Criar um snapshot de documentação \(p. 499\)](#)
- [Atualizar um snapshot de documentação \(p. 499\)](#)
- [Obter um snapshot de documentação \(p. 500\)](#)

- [Associar um snapshot de documentação a um estágio de API \(p. 500\)](#)
- [Baixar um snapshot de documentação associado a um estágio \(p. 501\)](#)

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: YYYYMMDDTTtttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
    "documentationVersion" : "1.0.0",
    "stageName": "prod",
    "description" : "My API Documentation v1.0.0"
}
```

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 novo 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: YYYYMMDDTTtttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
    "documentationVersion" : "1.0.0",
    "description" : "My API Documentation v1.0.0"
}
```

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: YYYYMMDDTTtttttZ
Authorization: AWS4-HMAC-SHA256 Credential=access_key_id/YYYYMMDD/region/
apigateway/aws4_request, SignedHeaders=content-length;content-type;host;x-amz-date,
Signature=sigv4_secret

{
    "patchOperations": [
        {
            "op": "replace",
            "path": "/description",
            "value": "My API for testing purposes."
        }
}
```

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"
        }
]
```

}

Baixar um snapshot de documentação associado a um estágio

Depois que uma versão das partes da documentação é associada a um estágio, você pode exportar essas 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 seus SDKs ou a AWS CLI do 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 `consultaeextension=documentation,integrations,authorizers` para incluir as partes de documentação da API, integrações da API e 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 contêm as propriedades de documentação com suporte pelo OpenAPI. A saída não inclui detalhes de integração nem dos autorizadores 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": {
        "summary": "Get the main page"
      }
    }
  }
}
```

```
"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: 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
```

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: 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
```

Para usar o console do API Gateway para exportar e fazer download da documentação publicada de uma API, siga as instruções em [Exportar a API REST usando o console do API Gateway \(p. 703\)](#).

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 \(p. 613\)](#) 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.

Importando 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": {}
            }
        ]
    }
}
```

```
        "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.

Importando 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. Escolha Documentation (Documentação) para a API, no painel de navegação principal do console.
2. Escolha Import Documentation (Importar documentação) no painel Documentation (Documentação).
3. Escolha Select OpenAPI File (Selecionar arquivo OpenAPI) para carregar um arquivo de uma unidade ou copiar e colar o conteúdo de um arquivo na exibição de arquivo. Para conhecer um exemplo, veja a carga da solicitação de exemplo em [Importando partes de documentação com a API REST do API Gateway \(p. 505\)](#).
4. Opcionalmente, escolha Fail on warnings (Falha nos avisos) ou Ignore warnings (Ignorar avisos), e escolha Merge ou Overwrite em Import mode (Modo de importação).
5. Escolha Import.

Controlar o acesso à documentação da API

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.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "StmtDocPartsAddEditViewDelete",
            "Effect": "Allow",
            "Action": [
                "apigateway:GET",
                "apigateway:PUT",
                "apigateway:POST",
                "apigateway:PATCH",
                "apigateway:DELETE"
            ],
            "Resource": [
                "arn:aws:apigateway:<region>:restapis/<restapi_id>/docs/partitions/<partition_id>"
            ]
        }
    ]
}
```

```
        "Resource": [
            "arn:aws:apigateway::account_id:restapis/*/documentation/*"
        ]
    }
}
```

Para obter informações sobre como definir permissões para acessar recursos do API Gateway, consulte [Controlar quem pode criar e gerenciar uma API do API Gateway com políticas IAM \(p. 380\)](#).

Atualizar e manter uma API REST no Amazon API Gateway

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 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 reimplantar a API, enquanto as atualizações de configuração, não.

Os recursos de API que podem ser atualizados são descritos na tabela a seguir.

Atualizações de recursos de API que exigem a reimplementação da API

Recurso	Observações
ApiKey	Para propriedades aplicáveis e operações compatíveis, acesse apikey:update . A atualização exige reimplantar a API.
Autorizador	Para propriedades aplicáveis e operações compatíveis, acesse authorizer:update . A atualização exige reimplantar a API.
DocumentationPart	Para propriedades aplicáveis e operações compatíveis, acesse documentationpart:update . A atualização exige reimplantar a API.
DocumentationVersion	Para propriedades aplicáveis e operações compatíveis, acesse documentationversion:update . A atualização exige reimplantar a API.
GatewayResponse	Para propriedades aplicáveis e operações compatíveis, acesse gatewayresponse:update . A atualização exige reimplantar a API.
Integration	Para propriedades aplicáveis e operações compatíveis, acesse integration:update . A atualização exige reimplantar a API.
IntegrationResponse	Para propriedades aplicáveis e operações compatíveis, acesse integrationresponse:update . A atualização exige reimplantar a API.
Método	Para propriedades aplicáveis e operações compatíveis, acesse method:update . A atualização exige reimplantar a API.
MethodResponse	Para propriedades aplicáveis e operações compatíveis, acesse methodresponse:update . A atualização exige reimplantar a API.
Modelo	Para propriedades aplicáveis e operações compatíveis, acesse model:update . A atualização exige reimplantar a API.
RequestValidator	Para propriedades aplicáveis e operações compatíveis, acesse requestvalidator:update . A atualização exige reimplantar a API.

Recurso	Observações
Recurso	Para propriedades aplicáveis e operações compatíveis, acesse resource:update . A atualização exige reimplantar a API.
RestApi	Para propriedades aplicáveis e operações compatíveis, acesse restapi:update . A atualização exige reimplantar a API.
VpcLink	Para propriedades aplicáveis e operações compatíveis, acesse vpclink:update . A atualização exige reimplantar a API.

As configurações de API que podem ser atualizadas são descritas na tabela a seguir.

Atualizações de configuração de API que não exigem a reimplantação da API

Configuração	Observações
Conta	Para propriedades aplicáveis e operações compatíveis, acesse account:update . A atualização não exige reimplantar a API.
Implantação	Para propriedades aplicáveis e operações compatíveis, acesse deployment:update .
DomainName	Para propriedades aplicáveis e operações compatíveis, acesse domainname:update . A atualização não exige reimplantar a API.
BasePathMapping	Para propriedades aplicáveis e operações compatíveis, acesse basepathmapping:update . A atualização não exige reimplantar a API.
Estágio	Para propriedades aplicáveis e operações compatíveis, acesse stage:update . A atualização não exige reimplantar a API.
Uso	Para propriedades aplicáveis e operações compatíveis, acesse usage:update . A atualização não exige reimplantar a API.
UsagePlan	Para propriedades aplicáveis e operações compatíveis, acesse usageplan:update . A atualização não exige reimplantar a API.

Tópicos

- [Alterar um tipo de endpoint de API pública ou privada no API Gateway \(p. 511\)](#)
- [Manter uma API usando o console do API Gateway \(p. 513\)](#)
- [Associar ou desassociar um VPC Endpoint de uma API REST privada \(p. 515\)](#)

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. Você pode alterar um tipo de API existente usando o console do API Gateway, a AWS CLI ou um AWS SDK para API Gateway. A operação de atualização pode levar até 60 segundos para ser concluída. Durante esse período, sua API estará disponível, mas o tipo de endpoint não pode ser alterado novamente até que a alteração atual seja concluída.

Há suporte para as seguintes alterações nos tipos de endpoints:

- De "otimizada para fronteiras" para "regional" ou "privada"

- De "regional" para "otimizada para fronteiras" 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 fronteiras" para "regional" ou vice-versa, observe que uma API otimizada para fronteiras 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 back-end 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 remeter o nome do cabeçalho para outro nome. A compreensão das diferenças ajuda você a decidir como atualizar uma API otimizada para fronteiras para uma regional ou de uma API regional para uma otimizada para fronteiras.

Tópicos

- [Usar o console do API Gateway para alterar um tipo de endpoint de API \(p. 512\)](#)
- [Use a AWS CLI para alterar um tipo de endpoint de API \(p. 512\)](#)

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 para otimizado para fronteiras e vice-versa

1. Faça login no console do API Gateway e escolha APIs no painel de navegação principal.
2. Escolha as configurações (ícone de engrenagem) de uma API em +Criar API.
3. Altere a opção Tipo de endpoint em Configuração de endpoint de Edge Optimized para Regional ou de Regional para Edge Optimized.
4. Escolha Salvar para iniciar a atualização.

Como converter um endpoint privado para um endpoint regional

1. Faça login no console do API Gateway e escolha APIs no painel de navegação principal.
2. Escolha as configurações (ícone de engrenagem) de uma API em +Criar API.
3. Edite a política de recursos da sua API para remover qualquer menção de VPCs ou VPC endpoints, a fim de que as chamadas de API de fora ou de dentro da sua VPC sejam bem-sucedidas.
4. Altere o Endpoint Type (Tipo de endpoint) para Regional.
5. Escolha Salvar para iniciar a atualização.
6. Remova a política de recursos da sua API.
7. Reimplante sua API para que as alterações sejam aplicadas.

Use a AWS CLI para alterar um tipo de endpoint de API

Para usar a AWS CLI para atualizar uma API otimizada para fronteiras cujo ID de API é `{api-id}`, chame `update-rest-api` da seguinte forma:

```
aws apigateway update-rest-api \
--rest-api-id {api-id}
--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": "0gsnjtjck8",  
    "name": "PetStore imported as edge-optimized"  
}
```

Por outro lado, atualize uma API regional para uma API otimizada para fronteiras da seguinte forma:

```
aws apigateway update-rest-api \  
  --rest-api-id {api-id} \  
  --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.

Manter uma API usando o console do API Gateway

Tópicos

- [Exibir uma lista de APIs no API Gateway \(p. 513\)](#)
- [Excluir uma API no API Gateway \(p. 513\)](#)
- [Excluir um recurso no API Gateway \(p. 514\)](#)
- [Exibir uma lista de métodos no API Gateway \(p. 514\)](#)
- [Excluir um método no API Gateway \(p. 514\)](#)

Exibir uma lista de APIs no API Gateway

Usar o console do API Gateway para exibir uma lista de APIs

Exibir uma lista de APIs com o console do API Gateway

Você deve ter uma API disponível no API Gateway. Siga as instruções em [Criação de uma API REST no Amazon API Gateway \(p. 182\)](#).

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. A lista de APIs é exibida.

Excluir uma API no API Gateway

Usar o console do API Gateway para excluir uma API.

Warning

Excluir uma API significa que você não pode mais chamá-la. Esta ação não pode ser desfeita.

Excluir uma API com o console do API Gateway

Você deve ter implantado a API pelo menos uma vez. Siga as instruções em [Implantando uma API REST no Amazon API Gateway \(p. 516\)](#).

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Na caixa que contém o nome da API que você deseja excluir, escolha Resources.
3. Escolha Delete API.
4. Quando solicitado a excluir a API, escolha Ok.

Excluir um recurso no API Gateway

Usar o console do API Gateway para documentar um recurso.

Warning

Quando você exclui um recurso, também exclui seus recursos e métodos filho. A exclusão de um recurso pode fazer com que parte da API correspondente se torne inutilizável. A exclusão de um recurso não pode ser desfeita.

Excluir um recurso com o console do API Gateway

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Na caixa que contém o nome da API do recurso que você deseja excluir, escolha Resources.
3. No painel Resources, escolha o recurso e escolha Delete Resource.
4. Quando solicitado, escolha Delete.

Exibir uma lista de métodos no API Gateway

Usar o console do API Gateway para exibir uma lista de métodos para um recurso

Exibir uma lista de métodos com o console do API Gateway

Você deve ter métodos disponíveis no API Gateway. Siga as instruções em [TUTORIAL: Criar uma API com integração não proxy HTTP \(p. 59\)](#).

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Na caixa que contém o nome da API, escolha Resources.
3. A lista de métodos é exibida no painel Resources.

Tip

Talvez você precise escolher a seta ao lado de um ou mais recursos para exibir todos os métodos disponíveis.

Excluir um método no API Gateway

Usar o console do API Gateway para excluir um método.

Warning

A exclusão de um método pode fazer com que parte da API correspondente se torne inutilizável. A exclusão de um método não pode ser desfeita.

Excluir um método com o console do API Gateway

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Na caixa que contém o nome da API para o método, escolha Recursos.
3. No painel Recursos, escolha a seta ao lado do recurso para o método.

4. Escolha o método e selecione Excluir método.
5. Quando solicitado, escolha Delete.

Associar ou desassociar um VPC Endpoint de uma API REST privada

Ao associar um VPC endpoint à sua API privada, o API Gateway gera um novo registro DNS de ALIAS do Route53 que pode ser usado para invocar suas APIs privadas da mesma forma que faz com suas APIs otimizadas para fronteiras ou regionais sem substituir um cabeçalho Host ou passar 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}
```

Associar ou desassociar um VPC endpoint a uma API REST privada exige que você atualize a configuração da API. É possível fazer essa alteração usando o console do API Gateway, a CLI da AWS ou um SDK da AWS para o API Gateway. A operação de atualização pode levar alguns minutos para ser concluída devido à propagação do DNS. Durante esse período, sua API estará disponível, mas a propagação de DNS para os URLs do DNS recém-gerados ainda poderá estar em andamento. É possível tentar [criar uma nova implantação para sua API \(p. 516\)](#), se seus novos URLs não estiverem resolvendo no DNS mesmo depois de alguns minutos.

Usar a CLI da AWS para associar um VPC endpoint a uma API REST privada

Para associar VPC endpoints no momento da criação da API, use o seguinte comando:

```
aws apigateway create-rest-api \
  --name Petstore \
  --endpoint-configuration '{ "types": [ "PRIVATE" ], "vpcEndpointIds" : [
    "vpce-0212a4ababd5b8c3e", "vpce-0393a628149c867ee" ] }' \
  --region us-west-2
{
  "apiKeySource": "HEADER",
  "endpointConfiguration": {
    "types": [
      "PRIVATE"
    ],
    "vpcEndpointIds": [
      "vpce-0212a4ababd5b8c3e",
      "vpce-0393a628149c867ee"
    ]
  },
  "id": "u67n3ov968",
  "createdDate": 1565718256,
  "name": "Petstore"
}
```

Para associar VPC endpoints a uma API privada já criada, use o seguinte comando da CLI:

```
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"  
}
```

Usar a CLI da AWS para desassociar um VPC endpoint de uma API REST privada

Para desassociar um VPC endpoint de uma API privada, use o seguinte comando da CLI:

```
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"  
}
```

Implantando uma API REST no Amazon 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. Cada estágio é um snapshot da API, sendo disponibilizado para ser chamado pelos aplicativos do cliente.

Important

Toda vez que atualiza uma API, o que inclui a modificação de rotas, métodos, integrações, autorizadores e qualquer outra atividade diferente das configurações de estágio, você deve reimplantar a API em um estágio novo ou já existente.

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 da versão canary \(p. 704\)](#), permitindo que os clientes da API acessem, no mesmo estágio, a versão de produção por meio do lançamento 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 a URL base da API.

Com o nome de domínio padrão da API, a URL base de uma (por exemplo) API REST em determinado estágio (`{stageName}`) tem o seguinte formato:

```
https://{restapi-id}.execute-api.{region}.amazonaws.com/{stageName}
```

Para facilitar o uso da URL base padrão de uma 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 desempenho 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 log de chamadas de API no CloudTrail ou no CloudWatch e selecionar um certificado de cliente para o back-end autenticar as solicitações da API. Além disso, você pode substituir as configurações no nível do estágio para os métodos individuais e definir as variáveis de estágio para que transmitam 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`. Essa promoção pode ser feita por meio da reimplementação da API para o estágio `prod` ou da atualização do valor de uma [variável de estágio \(p. 519\)](#) do nome de estágio do `test` para o `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 para fazer o mesmo, consulte a documentação da CLI da AWS ou de um [SDK da AWS](#), entre outros.

Tópicos

- [Implantar uma API REST no API Gateway \(p. 518\)](#)
- [Configurar um estágio no API Gateway \(p. 520\)](#)

- Gerar um SDK para uma API REST no API Gateway (p. 548)

Implantar uma API REST no API Gateway

No API Gateway, uma implantação de API REST é representada por um recurso [Implantação](#). Ela é como um executável de uma API que é representado 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 [Estágio](#) e representa um snapshot da API, incluindo métodos, integrações, modelos, modelos de mapeamento, autorizadores do Lambda (anteriormente conhecidos como autorizadores personalizados) etc. 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” \(p. 520\)](#).

Tópicos

- [Criar uma implantação usando a AWS CLI \(p. 518\)](#)
- [Implantar uma API REST no console do API Gateway \(p. 518\)](#)

Criar uma implantação usando a AWS CLI

A criação de uma implantação equivale a instanciar o recurso [Implantação](#). Você pode usar o console do API Gateway, a AWS CLI, um SDK da AWS ou a API REST do API Gateway para criar uma implantação.

Para usar a CLI a fim de criar uma implantação, use o comando `create-deployment`:

```
aws apigateway create-deployment --rest-api-id <rest-api-id> --region <region>
```

A API não poderá ser chamada até que esta implantação seja associada a um estágio. Com um estágio já existente, você pode fazer isso atualizando a propriedade [deploymentId](#) do estágio com o ID de implantação recém-criado ([<deployment-id>](#)).

```
aws apigateway update-stage --region <region> \  
  --rest-api-id <rest-api-id> \  
  --stage-name <stage-name> \  
  --patch-operations op='replace',path='/deploymentId',value='<deployment-id>'
```

Ao implantar uma API pela primeira vez, você pode combinar a criação do estágio e da implantação ao mesmo tempo:

```
aws apigateway create-deployment --region <region> \  
  --rest-api-id <rest-api-id> \  
  --stage-name <stage-name>
```

Isso é feito, nos bastidores, no console do API Gateway quando você implanta uma API pela primeira vez ou quando reimplanta a API em um novo estágio.

Implantar uma API REST no console do API Gateway

Você deve ter criado uma API REST antes de implantá-la pela primeira vez. Para obter mais informações, consulte [Criação de uma API REST no Amazon API Gateway \(p. 182\)](#).

Tópicos

- [Implantar uma API REST em um estágio \(p. 519\)](#)
- [Reimplantar uma API REST em um estágio \(p. 519\)](#)
- [Atualizar a configuração de estágio de uma implantação da API REST \(p. 519\)](#)

- [Definir variáveis de estágio para a implantação de uma API REST \(p. 519\)](#)
- [Associar um estágio à implantação de uma API REST diferente \(p. 520\)](#)

Implantar uma API REST em um estágio

O console do API Gateway permite que você implante uma API, criando uma implantação e associando-a a um estágio novo ou existente.

Note

Para associar um estágio no API Gateway a uma implantação diferente, consulte [Associar um estágio à implantação de uma API REST diferente \(p. 520\)](#) em vez disso.

1. Faça login 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 de navegação Recursos, escolha Ações.
4. No menu suspenso Ações, escolha Implantar API.
5. Na caixa de diálogo Implantar API, escolha uma entrada na lista suspensa Estágio de implantação.
6. Se você escolher [Novo estágio], digite um nome em Nome do estágio e, opcionalmente, forneça uma descrição para a etapa e a implantação em Descrição do estágio e Descrição da implantação. Se você escolher um estágio existente, talvez queira fornecer uma descrição da nova implantação em Descrição da implantação.
7. Escolha Deploy (Implantar) para implantar a API no estágio especificado com as configurações de estágio padrão.

Reimplantar uma API REST em um estágio

Para reimplantar uma API, execute as mesmas etapas descritas em [the section called “Implantar uma API REST em um estágio” \(p. 519\)](#). É possível reutilizar o mesmo estágio quantas vezes quiser.

Atualizar a configuração de estágio de uma implantação da API REST

Depois que uma API é implantada, você pode modificar as configurações de estágio para habilitar ou desabilitar o cache, o registro em log ou o controle de fluxo de solicitações dessa API. Você também pode escolher um certificado de cliente para o back-end 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 obter mais informações, consulte [Atualizar configurações de estágio \(p. 521\)](#).

Important

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 logs, exigirem uma nova função IAM, você poderá adicionar a função IAM necessária sem redistribuir a API. No entanto, pode demorar alguns minutos antes que a nova função IAM entre em vigor. Antes que isso aconteça, os rastreamentos das suas chamadas de API não serão registrados, mesmo que você tenha habilitado a opção de log.

Definir variáveis de estágio para a implantação de uma API REST

Para uma implantação, você pode 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 [Configurar variáveis de estágio para a implantação de uma API REST \(p. 540\)](#).

Associar um estágio à implantação de uma API REST diferente

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 Editor de estágio no console do API Gateway. Supõe-se que você tenha implantado uma API mais de uma vez.

1. Caso ainda não tenha feito isso, em Editor de estágio, escolha o estágio cuja implantação você deseja atualizar na opção Estágios de uma API no painel de navegação principal APIs.
2. Na guia Histórico de implantação, escolha o botão de opção próximo da implantação a ser usada pelo estágio.
3. Escolha Alterar implantação.

Configurar um estágio no API Gateway

Um estágio é uma referência designada para uma implantação, que é um snapshot da API. Você usa um [Estágio](#) 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 de logs, definir variáveis do estágio ou anexar uma versão canary para testes.

Tópicos

- [Configurar um estágio usando o console do API Gateway \(p. 520\)](#)
- [Habilitar o armazenamento em cache de APIs para melhorar a capacidade de resposta \(p. 525\)](#)
- [Controlar o fluxo de solicitações de API para uma melhor produtividade \(p. 531\)](#)
- [Configurar a autenticação SSL do lado do cliente no API Gateway \(p. 533\)](#)
- [Configurar o AWS WAF no API Gateway \(p. 534\)](#)
- [Configurar tags para um estágio da API no API Gateway \(p. 534\)](#)
- [Configurar registro de API em logs do CloudWatch no API Gateway \(p. 536\)](#)
- [Configurar o rastreamento do X-Ray no API Gateway \(p. 539\)](#)
- [Configurar o registro de acesso em logs ao Amazon Kinesis Data Firehose no API Gateway \(p. 540\)](#)
- [Configurar registro em logs do CloudTrail no API Gateway \(p. 540\)](#)
- [Configurar variáveis de estágio para a implantação de uma API REST \(p. 540\)](#)

Configurar um estágio usando o console do API Gateway

Tópicos

- [Criar um novo estágio \(p. 520\)](#)
- [Atualizar configurações de estágio \(p. 521\)](#)
- [Excluir um estágio para uma API \(p. 524\)](#)

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 e usar um novo 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 fazer isso usando o console do API Gateway, siga as instruções abaixo.

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel de navegação APIs, escolha Stages (Estágios) em uma API.
3. No painel de navegação Stages (Estágios), escolha Create (Criar).
4. Em Create Stage (Criar estágio), digite um nome de estágio, por exemplo, **prod**, para Stage name (Nome do estágio).

Note

Nomes de estágio podem conter apenas caracteres alfanuméricos, hífens e sublinhados. O tamanho máximo é de 128 caracteres.

5. Opcionalmente, digite uma descrição do estágio para Stage description (Descrição do estágio)
6. Na lista suspensa Deployment (Implantação), escolha a data e a hora da implantação de API existente que você deseja associar a esse estágio.
7. Escolha Criar.

Atualizar configurações de estágio

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. As etapas a seguir mostram como fazer isso usando o Stage Editor (Editor de estágio) do console do API Gateway.

Atualizar configurações de estágio usando o console do API Gateway

Estas instruções presumem que você já tenha implantado a API em um estágio.

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. No painel APIs, escolha a API e, em seguida, Stages (Estágios).
3. No painel Stages (Estágios), escolha o nome do estágio.
4. No painel Stage Editor (Editor de estágio), escolha a guia Settings (Configurações).
5. Para ativar o armazenamento em cache da API para o estágio, selecione a opção Enable API cache (Habilitar cache da API), na seção Cache Settings (Configurações de cache). Depois, escolha as opções desejadas e os valores associados para Cache capacity (Capacidade de cache), Encrypt cache data (Criptografar dados do cache), Cache time-to-live (TTL) (Tempo de vida (TTL) do cache), bem como os requisitos para invalidação de cache por chave.

Para obter mais informações sobre configurações de cache em nível de estágio, consulte [Habilitar o armazenamento em cache de APIs \(p. 525\)](#).

Important

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.

Tip

Você também pode substituir configurações de cache habilitadas em nível de estágio por métodos individuais. Para fazer isso, expanda o estágio no painel de navegação secundário Stages (Estágios) e escolha um método. Em seguida, no editor de estágio, escolha a opção Override for this method (Substituir para este método) em Settings (Configurações). Na área Cache Settings (Configurações de cache), marque ou desmarque Enable Method

Cache (Habilitar cache de método), ou personalize as opções desejadas. Para obter mais informações sobre as configurações de cache em nível de método, consulte [Habilitar o armazenamento em cache de APIs \(p. 525\)](#).

6. Para habilitar o Amazon CloudWatch Logs para todos os métodos associados a esse estágio dessa API do API Gateway, faça o seguinte:
 - a. Na seção CloudWatch Settings (Configurações do CloudWatch), selecione a opção Enable CloudWatch Logs (Habilitar o CloudWatch Logs).

Tip

Para habilitar as configurações no nível do método do CloudWatch, expanda o estágio no painel de navegação secundário Stages (Estágios) e escolha cada método de interesse. De volta ao editor de estágio, escolha Override for this method (Substituir para este método) em Settings (Configurações). Na área CloudWatch Settings (Configurações do CloudWatch), selecione Log to CloudWatch Logs (Registrar no CloudWatch Logs) e quaisquer outras opções desejadas, antes de escolher Save Changes (Salvar alterações).

Important

Sua conta será cobrada pelo acesso a métricas do CloudWatch em nível de método, mas não a métricas em nível de API ou estágio.

- b. Em Log level (Nível de log), selecione ERROR para gravar somente entradas em nível de erro no CloudWatch Logs, ou selecione INFO para incluir todos os eventos ERROR, bem como eventos informativos extras.
- c. Para registrar solicitações de chamadas de API completas e as informações de resposta, selecione Log full requests/responses data (Registrar dados completos de solicitações/respostas). Nenhum dado confidencial será registrado, a menos que a opção Log full requests/responses data (Registrar dados completos de solicitações/respostas) esteja selecionada.

Important

A configuração de logs como ERROR (ERRO) e a seleção de Log full requests/responses data (Registrar dados completos de solicitações/respostas) fazem com que todas as solicitações sejam registradas em detalhes. Esse é o comportamento planejado.

- d. Para que o API Gateway relate ao CloudWatch as métricas da API de API calls, Latency, Integration latency, 400 errors e 500 errors, escolha a opção Enable Detailed CloudWatch Metrics (Ativar as métricas detalhadas do CloudWatch). Para obter mais informações sobre o CloudWatch, consulte o [Guia do usuário do Amazon CloudWatch](#).
- e. Escolha Save Changes (Salvar alterações). As novas configurações entrarão em vigor após uma nova implantação.

Important

Para habilitar o CloudWatch Logs para todos os métodos, ou apenas alguns deles, você também deve especificar o ARN de uma função do IAM que permita ao API Gateway gravar informações no CloudWatch Logs em nome do seu usuário do IAM. Para isso, escolha Settings (Configurações) no painel de navegação principal APIs. Depois, insira o ARN de uma função do IAM no campo de texto CloudWatch log role ARN (ARN da função de log do CloudWatch). Para cenários de aplicativos comuns, a função do IAM poderá anexar a política gerenciada de `AmazonAPIGatewayPushToCloudWatchLogs`, que contém a seguinte instrução de política de acesso:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "logs:PutLogEvents",  
            "Resource": "arn:aws:logs:  
                <region>:  
                <account>/aws/apiGateway/  
                <stage>/  
                <method>/  
                <httpVerb>"  
        }  
    ]  
}
```

```
        "Action": [
            "logs>CreateLogGroup",
            "logs>CreateLogStream",
            "logs>DescribeLogGroups",
            "logs>DescribeLogStreams",
            "logs>PutLogEvents",
            "logs>GetLogEvents",
            "logs>FilterLogEvents"
        ],
        "Resource": "*"
    }
}
```

A função IAM também deve conter a seguinte instrução de relação de confiança:

```
{
    "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](#).

7. Para habilitar o registro de acesso em logs a um fluxo de entrega do Kinesis Data Firehose:
 - a. Em Custom Access Logging (Registro de acesso personalizado em logs), selecione Enable Access Logging (Habilitar o registro de acesso em logs).
 - b. Em Access Log Destination ARN (ARN do destino do log de acesso), insira o ARN de um fluxo de entrega do Kinesis Data Firehose. O formato do ARN é:
`arn:aws:firehose:{region}:{account-id}:deliverystream:amazon-apigateway-{your-delivery-stream-name}`.

Note

O nome do fluxo de entrega do Kinesis Data Firehose deve ser `amazon-apigateway-{your-delivery-stream-name}`.

- c. Em Log Format (Formato do log), insira um formato de log. Para usar um dos exemplos fornecidos como um guia, é possível escolher CLF, JSON, XML ou CSV.
- d. Escolha Save Changes (Salvar alterações).
8. Para definir limitações em nível de estágio para todos os métodos associados a essa API, faça o seguinte na seção Default Method Throttling (Limite de método padrão):
 - a. Selecione Enable throttling (Habilitar limitação).
 - b. Em Rate (Taxa), insira o número máximo de solicitações de estado fixo em nível de estágio por segundo que o API Gateway pode atender sem retornar uma resposta 429 Too Many Requests. Esse limite de taxas em nível de estágio não deve ser maior que o limite de taxas em nível de conta (p. 532), conforme especificado em [Limites do API Gateway para configurar e executar uma API REST \(p. 726\)](#).
 - c. Em Burst (Intermitência), digite o número máximo de solicitações simultâneas em nível de estágio que o API Gateway pode atender sem retornar uma resposta 429 Too Many Requests. Essa

intermitência em nível de estágio não deve ser maior que o limite de intermitência em [nível de conta \(p. 532\)](#), conforme especificado em [Limites do API Gateway para configurar e executar uma API REST \(p. 726\)](#).

9. Para substituir a limitação em nível de estágio para um método específico, expanda o estágio no painel de navegação secundário Stages (Estágios), escolha um método e, em seguida, escolha Override for this method (Substituir para este método) em Settings (Configurações). Na seção Method Throttling (Controle de utilização do método), selecione as opções apropriadas.
10. Para associar uma ACL da web do AWS WAF com o estágio, selecione uma ACL da web da lista suspensa de ACL da web.

Note

Se necessário, selecione Criar ACL da web para abrir o console do AWS WAF em uma nova guia do navegador, crie a ACL da web e retorne ao console do API Gateway para associar a ACL da web ao estágio.

11. Se desejar, selecione Bloquear solicitações de API se não for possível avaliar a WebACL (Falha - Fechar).
12. Para habilitar o rastreamento do [AWS X-Ray](#) para o estágio da API:
 - a. No painel Stage Editor (Editor de estágio), escolha a guia Logs/Tracing (Registros/Rastreamento).
 - b. Para habilitar o rastreamento do X-Ray, selecione Enable X-Ray Tracing (Habilitar o rastreamento do X-Ray) em X-Ray Tracing (Rastreamento do X-Ray).
 - c. Para definir regras de amostragem no console do X-Ray, escolha Set X-Ray Sampling Rules (Definir regras de amostragem do X-Ray).
 - d. Se quiser, escolha Set X-Ray Sampling Rules (Definir regras de amostragem do X-Ray) e acesse o console do X-Ray para [configurar regras de amostragem](#).

Para obter mais informações, consulte [Rastrear execução da API do API Gateway com o AWS X-Ray \(p. 587\)](#).

13. Escolha Save Changes (Salvar alterações). As novas configurações serão aplicadas depois que você implantar novamente a API no estágio.

Excluir um estágio para uma API

Quando você não precisar mais de um estágio, poderá excluí-lo para evitar pagar por recursos não utilizados. A seguir, explicamos como usar o console do API Gateway para excluir um estágio .

Warning

A exclusão de um estágio pode fazer com que parte da API correspondente, ou toda ela, se torne inutilizável pelos agentes de chamadas de API. Esse processo não pode ser desfeito, mas você pode recriar o estágio e associá-lo à mesma implantação.

Excluir um estágio com o console do API Gateway

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Na caixa que contém o nome da API para o estágio, escolha Estágios.
3. No painel Estágios, escolha o estágio que você deseja excluir e escolha Excluir estágio.
4. Quando solicitado, escolha Delete.

Habilitar o armazenamento em cache de APIs para melhorar a capacidade de resposta

Você pode habilitar o armazenamento em cache de APIs no Amazon API Gateway para armazenar em cache as respostas do seu 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. Ao habilitar o armazenamento em cache para um estágio, o API Gateway armazena em cache as respostas do seu endpoint por um período de vida útil (TTL) especificado, em segundos. Em seguida, 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.

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 \(p. 526\)](#).

Important

O armazenamento em cache é cobrado por hora e não está qualificado para o nível gratuito da AWS.

Habilitar o armazenamento em cache do Amazon API Gateway

No API Gateway, você pode habilitar o armazenamento em cache para todos um estágio especificado.

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.

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.

No console do API Gateway, configure o armazenamento em cache na guia Settings (Configurações) de um Stage Editor (Editor de estágio) denominado.

Para configurar o armazenamento em cache de API para um determinado estágio:

1. Acesse o console do API Gateway.
2. Selecione a API.
3. Escolha Stages (Estágios).
4. Na lista Stages (Estágios) da API, escolha o estágio.
5. Escolha a guia Configurações.
6. Escolha Enable API cache (Habilitar cache da API).
7. Aguarde a conclusão da criação do cache.

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 Cache status (Status de cache) é alterado de CREATE_IN_PROGRESS para AVAILABLE. Quando a exclusão do cache é concluída, o valor de Cache status (Status de cache) muda de DELETE_IN_PROGRESS para uma string vazia.

Ao habilitar o armazenamento em cache dentro das Cache Settings (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 \(p. 526\)](#).

Se quiser verificar se o armazenamento em cache está funcionando como esperado, você tem duas opções gerais:

- Inspecione as métricas do CloudWatch de CacheHitCount e CacheMissCount para a API e para o estágio.
- Colocar um carimbo de data/hora na resposta.

Note

Você não deve usar 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 em nível de estágio do API Gateway para o armazenamento em cache de métodos](#)

Você pode substituir as configurações de cache em nível de estágio ao habilitar ou desabilitar o armazenamento em cache para um método específico, aumentando ou diminuindo o seu período de TTL, ou ativando ou desativando a criptografia para respostas em cache.

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. Acesse o console do API Gateway.
3. Selecione a API.
4. Escolha Stages (Estágios).
5. Na lista Stages (Estágios) da API, expanda o estágio e escolha um método na API.
6. Escolha Override for this method (Sobreposição para esse método) em Settings (Configurações).
7. Na área Cache Settings (Configurações de cache), desmarque Enable Method Cache (Habilitar cache de método) ou personalize as opções desejadas. (Esta seção é mostrada somente se o [armazenamento em cache em nível de estágio \(p. 525\)](#) estiver ativado).

[Usar parâmetros de método ou integração como chaves de cache para indexar respostas em cache](#)

Quando um método ou uma integração em cache tem parâmetros, que podem assumir a forma de cabeçalhos personalizados, caminhos de URL ou strings de consulta, você pode usar alguns ou todos os parâmetros para formar chaves de cache. O API Gateway pode armazenar as respostas do método em cache, dependendo dos valores de parâmetros usados.

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 de 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 será armazenada em cache e usada para atender à seguinte solicitação:

```
GET /users?type=admin&department=B HTTP/1.1
host: example.com
...
```

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.

[Method Execution](#) /streams - GET - Method Request

Provide information about this method's authorization settings and the parameters it can receive.

Authorization Settings •

Authorization NONE

API Key Required false

▼ URL Query String Parameters •

Name	Caching	
query	<input checked="" type="checkbox"/>	

Add query string

► HTTP Request Headers

► Request Models [Create a Model](#) •

Descarregar o cache de estágios de API no API Gateway

Quando o armazenamento em cache de APIs está habilitado, você pode descarregar o cache inteiro do estágio de API para garantir que os clientes da sua API obtenham as respostas mais recentes dos seus endpoints de integração.

Para descarregar o cache do estágio da API, você pode escolher o botão Flush entire cache (Descarregar todo o cache) na seção Cache Settings (Configurações de cache) na guia Settings (Configurações) em um editor de estágio do console do API Gateway. A operação de descarga do cache demora alguns minutos, depois dos quais o status de cache será AVAILABLE imediatamente após a liberação.

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.

Invalidar 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.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "execute-api:InvalidateCache"  
            ],  
            "Resource": [  
                "arn:aws:execute-api:region:account-id:api-id/stage-name/GET/resource-path-specifier"  
            ]  
        }  
    ]  
}
```

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 com permissões do IAM \(p. 378\)](#)

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 é habilitado e uma autorização é necessária. É possível controlar como as solicitações não autorizadas são tratadas, escolhendo uma opção em `Handle unauthorized requests` (Tratar solicitações não autorizadas) no console do API Gateway.

test Stage Editor

● Invoke URL: https://██████████b.execute-api.us-east-1.amazonaws.com/test

Settings **Stage Variables** **SDK Generation** **Export** **Deployment History**

Configure the metering and caching settings for the **test** stage.

Cache Settings

Cache status AVAILABLE [Flush entire cache](#)

Enable API cache

Enabling API cache increases cost and is not covered by the free tier. See pricing for more details

Cache capacity 0.5GB ▾

Encrypt cache data

Cache time-to-live (TTL) 300 ▾

Per-key cache invalidation

Require authorization

Handle unauthorized requests Ignore cache control header; Add a warning in response header ▾

CloudWatch Settings	Ignore cache control header Add a warning in response header Ignore cache control header Fail the request with 403 status code
---------------------	---

Enable CloudWatch Logs ⓘ

As três opções resultam nos seguintes comportamentos:

- Fail the request with 403 status code (Falha na solicitação com o código de status 403): retorna uma resposta não autorizada 403

Para definir essa opção usando a API, use FAIL WITH 403.

- Ignore cache control header; Add a warning in response header (Ignorar cabeçalho de controle de cache; Adicionar um aviso ao cabeçalho de resposta): processa a solicitação e inclui um cabeçalho de aviso na resposta.

Para definir essa opção usando a API, use `SUCCEED_WITH_RESPONSE_HEADER`.

- Ignore cache control header (Ignorar cabeçalho de controle de cache): processa a solicitação e não inclui um cabeçalho de aviso na resposta.

Para definir essa opção usando a API, use `SUCCEED WITHOUT RESPONSE HEADER`.

Controlar o fluxo de solicitações de API para uma melhor produtividade

Para impedir que a sua API seja sobrecarregada por tantas solicitações, o Amazon API Gateway controla o fluxo das solicitações para a sua API usando o [algoritmo de bucket de token](#), em que um token equivale a uma solicitação. 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. No algoritmo de bucket de token, a intermitência é o tamanho máximo do bucket.

Quando os envios de solicitações excederem os limites de intermitência e taxas de estado fixo, o API Gateway marcará como falhas as solicitações que excederem o limite e retornará respostas de erro `429 Too Many Requests` para o cliente. Após capturar essas exceções, o cliente pode reenviar as solicitações com falha de forma limitante em termos de taxa, respeitando ao mesmo tempo os limites de controle de fluxo do API Gateway.

Como desenvolvedor de APIs, você pode definir os limites para diferentes estágios ou métodos de API, para melhorar o desempenho geral em todas as APIs na sua conta. Como alternativa, você pode habilitar [planos de uso](#) (p. 450) para restringir envios de solicitações de clientes às cotas e taxas de solicitações especificadas. Isso restringe os envios de solicitações em geral, de forma que eles não ultrapassem significativamente os limites de controle de fluxo em nível de conta.

Tópicos

- [Como as configurações de limitação de uso são aplicadas no API Gateway \(p. 531\)](#)
- [Controle de fluxo em nível de conta \(p. 532\)](#)
- [Limitação de uso de método padrão e substituição da limitação de uso de método padrão \(p. 533\)](#)
- [Configuração de limitação de uso em nível de API e em nível de estágio em um plano de uso \(p. 533\)](#)
- [Configuração de limitação de uso em nível de método em um plano de uso \(p. 533\)](#)

Como as configurações de limitação de uso são aplicadas no API Gateway

Antes de configurar as configurações de limite para sua API em suas configurações de estágio e, opcionalmente, um [plano de uso](#) (p. 450), é útil compreender como são aplicadas as configurações de limitação de uso do Amazon API Gateway.

O Amazon API Gateway fornece dois tipos básicos de configuração relacionada à limitação de uso:

- As limitações de uso do lado do servidor são aplicadas a todos os clientes. Essas configurações de limite existem para impedir que sua API — e sua conta — seja sobrecarregada com muitas solicitações.
- As limitações de uso por cliente são aplicadas aos clientes que usam chaves de API associadas à sua política de uso como identificador de cliente.

As configurações relacionadas à limitação de uso do API Gateway são aplicadas na seguinte ordem:

1. [Limitações de uso por método e por cliente \(p. 533\)](#) que você define para um estágio de API em um [plano de uso](#) (p. 458)
2. [Limitações de uso por cliente \(p. 533\)](#) que você define em um plano de uso
3. [Limites por método padrão e limites para métodos específicos \(p. 533\)](#) que você define nas configurações de estágio de API (p. 521)
4. [Limitações de uso em nível de conta \(p. 532\)](#)

Controle de fluxo em nível de conta

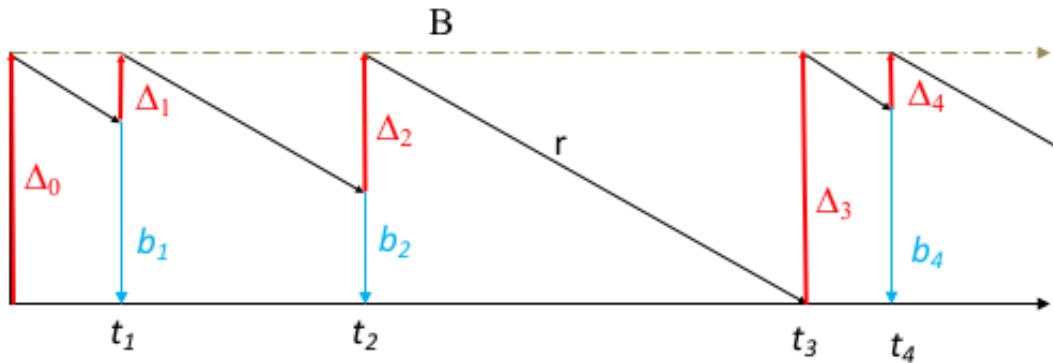
Por padrão, o API Gateway limita a taxa de solicitações em estado fixo a 10.000 solicitações por segundo (rps). Ele limita a intermitência (ou seja, o tamanho máximo do bucket) para 5.000 solicitações em todas as APIs de uma conta da AWS. No API Gateway, o limite de intermitência corresponde ao número máximo de envios simultâneos de solicitações que o API Gateway pode realizar a qualquer momento sem retornar respostas de erro 429 `Too Many Requests`.

Para ajudar a entender esses limites de controle, veja a seguir alguns exemplos, considerando o limite de intermitência e o limite da taxa no nível de conta padrão:

- Se um chamador enviar 10.000 solicitações em um período de um segundo uniformemente (por exemplo, 10 solicitações a cada milissegundo), o API Gateway processará todas as solicitações sem descartar nenhuma.
- Se o chamador enviar 10.000 solicitações no primeiro milissegundo, o API Gateway atenderá 5.000 dessas solicitações e limitará o restante no período de um segundo.
- Se o chamador enviar 5.000 solicitações no primeiro milissegundo e, em seguida, difundir uniformemente outras 5.000 solicitações durante os 999 milissegundos restantes (por exemplo, cerca de 5 solicitações a cada milissegundo), o API Gateway processará todas as 10.000 solicitações no período de um segundo sem retornar respostas de erro 429 `Too Many Requests`.
- Se o chamador enviar 5.000 solicitações no primeiro milissegundo e aguardar até o 101º milissegundo para enviar outras 5.000 solicitações, o API Gateway processará 6.000 solicitações e limitará o restante no período de um segundo. Isso ocorre porque a taxa de 10.000 rps, o API Gateway atendeu 1.000 solicitações depois dos primeiros 100 milissegundos e, portanto, esvaziou o bucket pelo mesmo valor. No próximo pico de 5.000 solicitações, 1.000 preenchem o bucket e são colocadas em fila para serem processadas. As outras 4.000 excedem a capacidade do bucket e são descartadas.
- Se o chamador enviar 5.000 solicitações no primeiro milissegundo, enviar 1.000 solicitações no 101º milissegundo e, em seguida, difundir uniformemente as 4.000 solicitações restantes durante os 899 milissegundos restantes, o API Gateway processará todas as 10.000 solicitações no período de um segundo, sem limitação.

De maneira mais genérica, em um determinado momento, quando um bucket contém b , e a capacidade máxima do bucket é B , o número máximo de tokens adicionais que podem ser adicionados ao bucket é $\# = B - b$. Esse número máximo de tokens adicionais corresponde ao número máximo de solicitações simultâneas adicionais que um cliente pode enviar sem receber respostas de erro 429. Em geral, $\#$ varia de acordo com o tempo. O valor varia de zero, quando o bucket está cheio (isto é, $b=B$), até B , quando ele está vazio (ou seja, $b=0$). O intervalo depende da taxa de processamento de solicitações, que é a taxa na qual os tokens são removidos do bucket, e da taxa limite, que é a taxa na qual os tokens são adicionados ao bucket.

O seguinte esquema mostra os comportamentos gerais do $\#$, o número máximo de solicitações simultâneas adicionais, como uma função de tempo. O esquema pressupõe que os tokens no bucket são reduzidos em uma taxa combinada de r , a partir de um bucket vazio.



O limite da taxa no nível de conta pode ser aumentado por meio de uma solicitação. Para solicitar um aumento nos limites de controle de fluxo em nível de conta, entre em contato com o [AWS Support Center](#). Para obter mais informações, consulte [Limites do API Gateway \(p. 725\)](#).

Limitação de uso de método padrão e substituição da limitação de uso de método padrão

Você pode definir a limitação de uso de método padrão para substituir as limitações de uso de solicitações em nível de conta para um estágio específico ou para métodos específicos em sua API. As limitações de uso de método padrão são restrinvidas pelos limites de taxa em nível de conta, mesmo que você defina limitações de uso de método padrão mais altos que os limites em nível de conta.

Você pode definir limitações de uso de método padrão no console do API Gateway usando a configuração Default Method Throttling (Limitação de uso de método padrão) em Stages (Estágios). Para obter instruções sobre como usar o console, consulte [Atualizar configurações de estágio \(p. 521\)](#).

Você pode também definir as limitações de uso de método padrão chamando [Referências de API na V1 e V2 do API Gateway \(p. 724\)](#).

Configuração de limitação de uso em nível de API e em nível de estágio em um plano de uso

Em um [plano de uso \(p. 450\)](#), você pode definir uma limitação de uso de método padrão para todos os métodos no nível de API ou estágio em Create Usage Plan (Criar plano de uso), conforme mostrado em [Criar um plano de uso \(p. 458\)](#).

Configuração de limitação de uso em nível de método em um plano de uso

Você pode definir limitações de uso adicionais em nível de método em Usage Plans (Planos de uso), conforme mostrado em [Criar um plano de uso \(p. 458\)](#). No console do API Gateway, isso é definido especificando `Resource=<resource>, Method=<method>` na configuração Configure Method Throttling (Configurar limitação de uso de método). Por exemplo, no [caso de PetStore \(p. 59\)](#), você pode especificar `Resource=/pets, Method=GET`.

Configurar a autenticação SSL do lado do cliente no API Gateway

Você pode usar o API Gateway para gerar um certificado SSL e usar sua chave pública no back-end para verificar se as solicitações HTTP para seu sistema back-end são provenientes do API Gateway. Também é

possível habilitar a autenticação SSL do lado do cliente para uma API usando o console do API Gateway. Para obter mais informações, consulte [the section called “Configurar uma API para usar certificados SSL” \(p. 430\)](#).

Configurar o AWS WAF no API Gateway

Você pode usar o AWS WAF para proteger seus aplicativos web e APIs contra ataques configurando um conjunto de regras (conhecido como lista de controle de acesso da web ou ACL da web) que permite, bloqueia ou calcula solicitações da web com base nas regras de segurança da web personalizáveis e nas condições que você definir. Para obter mais informações, consulte [the section called “Use o AWS WAF para proteger sua API contra explorações comuns da web” \(p. 449\)](#).

Configurar tags para um estágio da 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/o 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 V1 do Amazon API Gateway” \(p. 719\)](#).

Para obter mais informações sobre a marcação de recursos do API Gateway, consulte [Marcar seus recursos do API Gateway \(p. 718\)](#).

Tópicos

- [Configurar tags para um estágio da API usando o console do API Gateway \(p. 534\)](#)
- [Configurar tags para um estágio da API usando a API REST do API Gateway \(p. 534\)](#)

Configurar tags para um estágio da API usando o console do API Gateway

O procedimento a seguir descreve como configurar tags para um estágio da API.

Para configurar tags para um estágio da API usando o console do API Gateway

1. Faça login 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 Stage Editor (Editor de estágio), escolha o botão Configure Tags (Configurar tags).
5. No Tag Editor (Editor de tag), escolha Add New Tag (Adicionar nova tag). Digite uma chave de tag (por exemplo, Department) na coluna Key (Chave) e digite um valor de tag (por exemplo, Sales) na coluna Value (Valor). Escolha o ícone de marca de seleção para salvar a tag.
6. Se necessário, repita a etapa 5 para adicionar mais tags ao estágio da API. O número máximo de tags por estágio é 50.
7. Para remover uma tag existente do estágio, escolha o ícone de lixeira ao lado da tag selecionada.
8. Escolha Save Changes (Salvar alterações) para terminar de configurar as tags do estágio.

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 API REST do API Gateway

Você pode configurar tags para um estágio da API usando a &API REST do API Gateway, seguindo uma das seguintes ações:

- Chame [tags:tag](#) para marcar um estágio da API.
- Chame [tags:untag](#) para excluir uma ou mais tags de um estágio da API.
- Chame [stage:create](#) para adicionar uma ou mais tags a um estágio da API que você estiver criando.

Você também pode chamar [tags:get](#) para descrever as tags em um estágio da 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%2Frestapis%2Fm5zr3vnks7%2Fstages%2Ftest`).

```
PUT /tags/arn%3Aaws%3Aapigateway%3Aus-east-1%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%2Frestapis%2Fm5zr3vnks7%2Fstages%2Ftest?tagKeys=Department
Host: apigateway.us-east-1.amazonaws.com
Authorization: ...
```

Para remover mais de uma tag, use uma lista separada por vírgulas de chaves de tag na expressão da consulta — por exemplo, `?tagKeys=Department,Division,...`.

Descrever tags para o estágio da API

Para descrever as tags existentes em um determinado estágio, chame [tags:get](#):

```
GET /tags/arn%3Aaws%3Apigateway%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%3Apigateway%3Aus-east-1%3A%3A%2Frestapis
%2Fm5zr3vnks7%2Fstages%2Ftags"
    },
    "tags:untag": {
      "href": "/tags/arn%3Aaws%3Apigateway%3Aus-east-1%3A%3A%2Frestapis
%2Fm5zr3vnks7%2Fstages%2Ftags{?tagKeys}",
      "templated": true
    }
  },
  "tags": {
    "Department": "Sales"
  }
}
```

Configurar registro de API em logs do CloudWatch no API Gateway

Para ajudar a depurar problemas relacionados à execução de solicitação ou ao acesso do cliente à sua API, você pode permitir que o Amazon CloudWatch Logs registre chamadas de API. Para obter mais informações sobre CloudWatch, consulte [the section called “Monitorar a execução da API com o Amazon CloudWatch” \(p. 599\)](#).

Formatos de logs do CloudWatch para o API Gateway

Há dois tipos de registro de API em logs na CloudWatch: registro de execução e de acesso. No registro de execução em logs, 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 chamador. Os dados registrados em logs incluem erros ou rastreamentos de execução (como cargas 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 assim por diante.

Quando você implanta uma API, o API Gateway cria um grupo de logs e registra os fluxos 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 chamador acessou a API. Você pode criar seu próprio grupo de logs ou escolher um existente, o qual pode ser gerenciado pelo API Gateway. Você pode especificar os detalhes de acesso selecionando `$context` (p. 306) variáveis, expressas em um formato de sua escolha e escolhendo um grupo de

logs como o destino. Para preservar a exclusividade de cada log, o formato do log de acesso deve incluir `$context.requestId`.

Note

Somente as variáveis `$context` têm suporte, não `$input`, etc.

Escolha um formato de log que também seja adotado pelo seu back-end 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 Nome da região da Amazon (ARN) do grupo de logs na propriedade `accessLogSettings/destinationArn` no [estágio](#). Você pode obter um Nome de região da Amazon (ARN) do grupo de logs no console do CloudWatch, desde que a coluna ARN esteja selecionada para exibição. 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
```

Os caracteres de continuação (\) servem de auxílio visual e o formato de log não pode ter quebras de linha.

- [JSON](#):

```
{ "requestId": "$context.requestId", \
  "ip": "$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" \
}
```

Os caracteres de continuação (\) servem de auxílio visual e o formato de log não pode ter quebras de linha.

- [XML](#):

```
<request id="$context.requestId"> \
  <ip>$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>
```

Os caracteres de continuação (\) servem de auxílio visual e o formato de log não pode ter quebras de linha.

- 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
```

Os caracteres de continuação (\) servem de auxílio visual e o formato de log não pode ter quebras de linha.

Permissões de log do CloudWatch

Para permitir o CloudWatch Logs, você deve conceder permissão de leitura e gravação de logs ao API Gateway no CloudWatch para a sua conta. A política gerenciada `AmazonAPIGatewayPushToCloudWatchLogs` (com um Nome de região da Amazon (ARN) de `arn:aws:iam::aws:policy/service-role/AmazonAPIGatewayPushToCloudWatchLogs`) tem todas as permissões necessárias:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "logs:CreateLogGroup",  
                "logs:CreateLogStream",  
                "logs:DescribeLogGroups",  
                "logs:DescribeLogStreams",  
                "logs:PutLogEvents",  
                "logs:GetLogEvents",  
                "logs:FilterLogEvents"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

Note

O API Gateway chamará o AWS Security Token Service para assumir a função do IAM, portanto, verifique se o AWS STS está 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 sua entidade confiável, anexe a política anterior à função do IAM e defina o Nome de região da Amazon (ARN) da função do IAM na propriedade `cloudWatchRoleArn` na sua conta. É necessário definir a propriedade `cloudWatchRoleArn` separadamente para cada região da AWS na qual você 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 AWS STS esteja habilitado na região que você está usando. Para obter mais informações sobre a habilitação do AWS STS, consulte [Gerenciar o AWS STS em uma região da AWS](#) no Guia do usuário do IAM.

Configurar o registro de API do CloudWatch em logs usando o console do API Gateway

Para configurar o registro de API do CloudWatch em logs, é necessário ter implantado a API em um estágio. Você também deve ter configurado [um Nome de região da Amazon \(ARN\) apropriado da função do CloudWatch Logs \(p. 538\)](#) para a sua conta.

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Selecione Settings (Configurações) no painel de navegação principal e digite um ARN de uma função do IAM com as permissões adequadas no CloudWatch log role ARN (ARN de função de registro do CloudWatch). Você precisa fazer isso apenas uma vez.
3. Siga um destes procedimentos:
 - a. Escolha uma API existente e, em seguida, escolha um estágio.
 - b. Crie uma API e implante-a em um estágio.
4. Escolha Logs/Tracing no Stage Editor.
5. Para habilitar o registro de execução em logs:
 - a. Selecione Enable CloudWatch Logs (Habilitar o CloudWatch Logs) em CloudWatch Settings (Configurações do CloudWatch).
 - b. Selecione Error (Erro) ou Info (Informações) no menu suspenso.
 - c. Se desejar, escolha Enable Detailed CloudWatch Metrics.

Para obter mais informações sobre métricas do CloudWatch, consulte o [the section called “Monitorar a execução da API com o Amazon CloudWatch” \(p. 599\)](#).

6. Para habilitar o registro de acesso em logs:
 - a. Selecione Enable Access Logging (Habilitar o registro de acesso em logs) em Custom Access Logging (Registro de acesso personalizado em logs).
 - b. Insira o ARN de um grupo de logs em Access Log Destination ARN (ARN do destino do log de acesso). O formato do ARN é `arn:aws:logs:{region}:{account-id}:log-group:API-Gateway-Execution-Logs_{rest-api-id}/{stage-name}`.
 - c. Insira um formato de log em Log Format (Formato do log). Você pode escolher CLF, JSON, XML ou CSV para usar um dos exemplos fornecidos como um guia.
7. Selecione Save Changes (Salvar alterações).

Note

Você pode permitir o registro de execução e de acesso em logs independentes um do outro.

O API Gateway já está pronto para registrar as solicitações a sua API em log. Você não precisa reimplantar a API ao atualizar as configurações de estágio, logs ou variáveis de estágio.

Configurar o rastreamento do X-Ray no API Gateway

Para ajudar a depurar problemas relacionados à latência de solicitação da API, habilite o rastreamento do AWS X-Ray para rastrear solicitações de API e serviços downstream. Quando habilitado, o API Gateway rastreará chamadas da API no X-Ray. Para obter mais informações, consulte [the section called “Configuração do AWS X-Ray” \(p. 587\)](#).

Configurar o registro de acesso em logs ao Amazon Kinesis Data Firehose no API Gateway

Você pode habilitar o registro de acesso em logs a um fluxo de entrega do Amazon Kinesis Data Firehose para auditoria e análise aprimoradas. Para obter mais informações, consulte [the section called “Usar o Kinesis Data Firehose com registro de acesso em logs do API Gateway” \(p. 605\)](#).

Configurar registro em logs do CloudTrail no API Gateway

Não há nada que você precise fazer para ativar o registro em log do CloudTrail para a API. Ele é ativado automaticamente. Para obter mais informações, consulte [the section called “Registrar chamadas para APIs do Amazon API Gateway com AWS CloudTrail” \(p. 597\)](#).

Configurar variáveis de estágio para a implantação de uma API REST

Variáveis de estágio são pares de nome/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.

Por exemplo, você pode definir uma variável de estágio em uma configuração de estágio e, em seguida, definir seu valor como a string de URL de uma integração HTTP para um método na sua API REST. Mais tarde, é possível fazer referência à string de URL usando o nome da variável de estágio associada da configuração da API. Dessa forma, você pode usar a mesma configuração de API com um endpoint diferente em cada estágio, redefinindo o valor da variável de estágio para as URLs correspondentes. Você também pode acessar variáveis de estágio em modelos de mapeamento ou passar parâmetros de configuração ao back-end HTTP ou AWS Lambda.

Para obter mais informações sobre modelos de mapeamento, consulte [Referência de variáveis de registro de acesso em logs e modelo de mapeamento do API Gateway \(p. 305\)](#).

Casos de uso

Com estágios de implantação no API Gateway, você pode gerenciar vários estágios de versão para cada API, como alfa, beta e produção. Usando variáveis de estágio, você pode configurar um estágio de implantação da API para interagir com diferentes endpoints de back-end. Por exemplo, sua API pode passar uma solicitação GET como um proxy HTTP ao host da Web do back-end (por exemplo, `http://example.com`). Nesse caso, o host da Web do back-end está configurado em uma variável de estágio de forma que, quando os desenvolvedores chamarem seu endpoint de produção, o API Gateway chamará `example.com`. Quando você chama seu endpoint beta, o API Gateway usa o valor configurado na variável de estágio para o estágio beta e chama um host da Web diferente (por exemplo, `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.

Você também pode usar variáveis de estágio para transmitir parâmetros de configuração para uma função Lambda por meio de seus modelos de mapeamento. Por exemplo, você pode querer reutilizar a mesma função Lambda para vários estágios na sua API, mas a função deve ler dados de uma tabela de Amazon DynamoDB diferente, dependendo do estágio no qual ela está sendo chamada. 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 para o Lambda.

As variáveis de estágio não são aplicadas à seção de definições de segurança da especificação da API. Por exemplo, você não pode usar grupos de usuários diferentes do Amazon Cognito para diferentes estágios.

Exemplos

Para usar uma variável de estágio para personalizar o endpoint de integração HTTP, você deve primeiro configurar uma variável de estágio com um nome específico, por exemplo, `url`, e depois atribuir a ela um valor, como `example.com`. Em seguida, na sua configuração de método, configure uma integração de proxy HTTP e, em vez de digitar 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 de qual estágio sua API está executando. Você pode referenciar variáveis de estágio de maneira semelhante, especificando um nome de função Lambda, um caminho de proxy de serviço da AWS ou um ARN de função da AWS no campo de credenciais.

Ao especificar um nome de função Lambda como um valor de variável de estágio, você deve configurar as permissões nessa função Lambda manualmente. Você pode usar o AWS Command Line Interface para fazer isso.

```
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
```

O exemplo a seguir atribui a permissão do API Gateway para invocar uma função do Lambda denominada `helloWorld`, hospedada na região Oeste dos EUA (Oregon) de uma conta da AWS em nome do método de API.

```
arn:aws:execute-api:us-west-2:123123123123:bmmuvptwze/*/*GET/hello
```

Este é o mesmo comando usando a AWS CLI.

```
aws lambda add-permission --function-name arn:aws:lambda:us-east-1:123123123123:function:helloWorld --source-arn arn:aws:execute-api:us-west-2:123123123123:bmmuvptwze/*/*GET/hello --principal apigateway.amazonaws.com --statement-id apigateway-access --action lambda:InvokeFunction
```

Definir variáveis de estágio usando o console do Amazon API Gateway

Neste tutorial, você aprenderá a configurar variáveis de estágio para dois estágios de implantação de uma API de amostra, usando o console do Amazon API Gateway. 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 [Criação de uma API REST no Amazon API Gateway \(p. 182\)](#).
- Você deve ter implantado a API pelo menos uma vez. Siga as instruções em [Implantando uma API REST no Amazon API Gateway \(p. 516\)](#).
- Você deve ter criado o primeiro estágio para uma API implantada. Siga as instruções em [Criar um novo estágio \(p. 520\)](#).

Para declarar variáveis de estágio usando o console do API Gateway

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Crie uma API e crie um método GET no recurso raiz da API, se ainda não tiver feito isso. Defina o valor Endpoint URL (URL de endpoint) HTTP como "`http://${stageVariables.url}`" e escolha Save (Salvar).
3. Escolha Deploy API (Implantar API). Escolha New Stage (Novo estágio) e digite "beta" para Stage name (Nome do estágio). Escolha Deploy (Implantar).

4. No painel beta Stage Editor (Editor de estágio beta), escolha a guia Stage Variables (Variáveis de estágio) e, em seguida, selecione Add Stage Variable (Adicionar variável de estágio).
5. Insira a string "url" no campo Name (Nome) e "httpbin.org/get" no campo Value (Valor). Escolha o ícone de marca de seleção para salvar a configuração da variável de estágio.
6. Repita a etapa acima para adicionar mais duas variáveis de estágio: `version` e `function`. Defina seus valores como "v-beta" e "HelloWorld", respectivamente.

Note

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 assume o valor da variável de estágio para uma função Lambda como o nome de função não qualificado e expandirá a variável de estágio especificada em um ARN.

7. No painel de navegação Stages (Estágios), escolha Create (Criar). Em Stage name (Nome do estágio), digite `prod`. Selecione uma implantação recente em Deployment (Implantação) e escolha Create (Criar).
8. Como no estágio beta, defina as mesmas três variáveis de estágio (`url`, `version` (versão) e `function` (função)) como valores diferentes ("`petstore-demo-endpoint.execute-api.com/petstore/pets`", "v-prod" e "HelloEveryone"), respectivamente.

Usar variáveis de estágio do Amazon API Gateway

É possível usar variáveis de estágio do API Gateway para acessar os back-ends HTTP e do Lambda para diferentes estágios de implantação da API e transmitir metadados de configuração específicos aos estágios para um back-end HTTP como um parâmetro de consulta e para uma função do Lambda como uma carga gerada em um modelo de mapeamento de entrada.

Pré-requisitos

Você deve criar dois estágios com uma variável `url` definida como dois endpoints HTTP: uma variável de estágio `function` atribuída a duas funções do Lambda diferentes e uma variável de estágio `version` que contém metadados específicos do estágio. Siga as instruções em [Definir variáveis de estágio usando o console do Amazon API Gateway \(p. 541\)](#).

Acessar um endpoint HTTP por meio de uma API com uma variável de estágio

1. No painel de navegação Stages (Estágios), escolha o estágio beta. Em Editor de estágio beta, escolha o link Invocar URL. 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. Navegar até a URL escolhendo o link chama o método GET de estágio beta no recurso raiz. Se métodos estiverem definidos em recursos filho, e não no próprio recurso raiz, escolher o link Invoke URL (Invocar URL) retornará 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).

2. 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.
3. No painel de navegação Stages (Estágios), escolha o nome do estágio prod. No prod Stage Editor (Editor de estágio prod), escolha o link Invoke URL (Invocar URL). Isso inicia a solicitação GET de estágio prod no recurso raiz da API.

4. 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.

[Passar metadados específicos ao estágio para um back-end HTTP por meio de uma variável de estágio em uma expressão de parâmetro de consulta](#)

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 back end HTTP. Usaremos a variável de estágio version declarada em [Definir variáveis de estágio usando o console do Amazon API Gateway \(p. 541\)](#).

1. No painel de navegação Resource (Recurso), escolha o método GET. Para adicionar um parâmetro de string de consulta ao URL do método, em Method Execution (Execução de método), escolha Method Request (Solicitação de método). Digite **version** para o nome do parâmetro.
2. Em Method Execution (Execução de método), escolha Integration Request (Solicitação de integração). Edite o valor Endpoint URL (URL de endpoint) para acrescentar **?version= \${stageVariables.version}** ao valor de URL previamente definido, que, neste caso, também é expresso com a variável de estágio url. Escolha Deploy API (Implantar API) para implantar essas alterações.
3. No painel de navegação Stages (Estágios), escolha o nome do estágio beta. Em beta Stage Editor (Editor de estágio beta), verifique se o estágio atual está na implantação mais recente e escolha o link Invoke URL (Invocar URL).

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 os retorna como o objeto args em sua resposta.

4. A resposta é mostrada a seguir. Observe que v-beta, atribuído à variável de estágio version, é passado para o back-end como o argumento version.

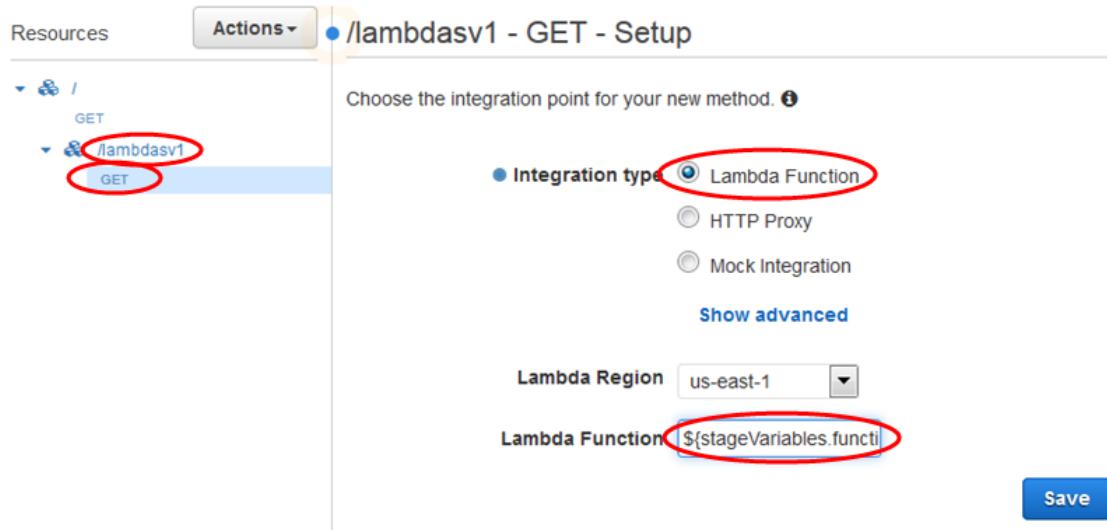
```
{  
  "args": {  
    "version": "v-beta"  
  },  
  "headers": {  
    "Accept": "application/json",  
    "Host": "httpbin.org",  
    "User-Agent": "AmazonAPIGateway_h4ah70cvmb"  
  },  
  "origin": "52.91.42.97",  
  "url": "http://httpbin.org/get?version=v-beta"  
}
```

[Chamar a função Lambda por meio da API com uma variável de estágio](#)

Este procedimento descreve como usar uma variável de estágio para chamar uma função Lambda como um back-end da sua API. Usaremos a variável de estágio function declarada anteriormente.

Para obter mais informações, consulte [Definir variáveis de estágio usando o console do Amazon API Gateway \(p. 541\)](#).

1. No painel Resources (Recursos), crie um recurso filho /lambdasv1 sob o diretório raiz e crie um método GET no recurso filho. Defina o Integration type (Tipo de integração) como Função Lambda e, em Lambda Function (Função Lambda), digite \${stageVariables.function}. Escolha Salvar.



Tip

Quando o sistema solicitar Add Permission to Lambda Function (Adicionar permissão à função do Lambda), anote o comando da CLI da AWS antes de selecionar OK. Você deve executar o comando em cada função do Lambda que está ou será atribuída à variável de estágio function para cada um dos métodos de API recém-criados. Por exemplo, se o valor \${stageVariables.function} for HelloWorld e você ainda não tiver adicionado permissão para essa função, deverá executar 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/lambdasv1 --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. Certifique-se de substituir \${stageVariables.function} pelo nome da função do Lambda atribuída à variável de estágio.



2. Implante a API nos estágios disponíveis.

3. No painel de navegação Stages (Estágios), escolha o nome do estágio beta. Verifique se a sua implantação mais recente está no Editor de estágio beta. Copie o link Invoke URL (Invocar URL), cole-o na barra de endereços do seu navegador e acrescente `/lambdasv1` a esse URL. Isso chama a função do Lambda subjacente por meio do método GET no recurso filho LambdaSv1 da API.

Note

Sua função `HelloWorld` do Lambda implementa o código a seguir.

```
exports.handler = function(event, context, callback) {
  if (event.version)
    callback(null, 'Hello, World! (' + event.version + ')');
  else
    callback(null, "Hello, world! (v-unknown)");
};
```

Essa implementação resulta na seguinte resposta.

```
"Hello, world! (v-unknown)"
```

Transmitir metadados específicos de estágio a uma função 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 Lambda. Usaremos um método POST e um modelo de mapeamento de entrada para gerar a carga usando a variável de estágio `version` declarada anteriormente.

1. No painel Resources (Recursos), escolha o recurso filho `/lambdasv1`. Crie um método POST no recurso filho, defina o Integration type (Tipo de integração) como Lambda Function (Função do Lambda) e digite `#{stageVariables.function}` em Lambda Function (Função do Lambda). Escolha Salvar.

Tip

Esta etapa é semelhante à etapa que usamos para criar o método GET. Para obter mais informações, consulte [Chamar a função Lambda por meio da API com uma variável de estágio \(p. 543\)](#).

2. No painel Method Execution (Execução de método), escolha Integration Request (Solicitação de integração). No painel Integration Request (Solicitação de integração), expanda Mapping Templates (Modelos de mapeamento) e escolha Add mapping template (Adicionar mapeamento de modelos) para adicionar um modelo para o tipo de conteúdo `application/json`, como mostrado no seguinte.

The screenshot shows the 'Method Execution' configuration for the POST method of the '/lambdasv1' resource. The 'Integration type' is set to 'Lambda Function'. The 'Lambda Region' is 'us-east-1'. The 'Lambda Function' is '\$(stageVariables.function)'. The 'Invoke with caller credentials' checkbox is unchecked. The 'Credentials cache' setting is 'Do not add caller credentials to cache key'. Under 'Body Mapping Templates', there is a mapping for 'Content-Type: application/json' with the value 'application/json'. A mapping template is shown in the code editor:

```
1 #set($inputRoot = $input('$'))  
2 {  
3     "version": "${stageVariables.version}"  
4 }
```

Note

Em um modelo de mapeamento, uma variável de estágio deve ser referenciada entre aspas (como em "\$stageVariables.version" ou "\${stageVariables.version}"), enquanto em outro lugar ela deve ser referenciada sem aspas (como em \${stageVariables.function}).

3. Implante a API nos estágios disponíveis.
4. No painel de navegação Stages (Estágios), escolha o estágio beta. Em beta Stage Editor (Editor de estágio beta), verifique se o estágio atual possui a implantação mais recente. Copie o link Invoke URL (Invocar URL), cole-o no campo de entrada de URL de um cliente da API REST, acrescente /lambdasv1 a esse URL e envie uma solicitação POST à função do Lambda subjacente.

Note

Você receberá a seguinte resposta.

```
"Hello, world! (v-beta)"
```

Referência a variáveis de estágio do Amazon 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 uma URL de integração HTTP, como mostram os exemplos a seguir.

- Um URI completo sem protocolo, por exemplo, `http://${stageVariables.<variable_name>}`
- Um domínio completo: por exemplo, `http://${stageVariables.<variable_name>}/resource/operation`
- Um subdomínio: por exemplo, `http://${stageVariables.<variable_name>.example.com/resource/operation}`
- Um caminho, por exemplo, `http://example.com/${stageVariables.<variable_name>}/bar`
- Uma string de consulta, por exemplo, `http://example.com/foo?q=${stageVariables.<variable_name>}`

URIs de integração da AWS

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>}`

URIs de integração da AWS (funções do Lambda)

Uma variável de estágio pode ser usada no lugar de um nome de função 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::<account_id>:function:${stageVariables.<function_variable_name>}/invocations`
- `arn:aws:apigateway:<region>:lambda:path/2015-03-31/functions/arn:aws:lambda::<account_id>:function:<function_name>:${stageVariables.<version_variable_name>}/invocations`

Credenciais de integração da AWS

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>}`

Gerar um SDK para uma API 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. Atualmente, o API Gateway oferece suporte para a 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 e 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 \(p. 554\)](#) do API Gateway, que expõe a função de [Calculadora simples \(p. 552\)](#) do Lambda.

Antes de prosseguir, crie ou importe a API e implante-a pelo menos uma vez no API Gateway. Para obter instruções, consulte [Implantando uma API REST no Amazon API Gateway \(p. 516\)](#).

Tópicos

- [Gerar SDKs para uma API usando o console do API Gateway \(p. 548\)](#)
- [Gerar SDKs para uma API usando comandos da AWS CLI \(p. 551\)](#)
- [Função Lambda da calculadora simples \(p. 552\)](#)
- [API de calculadora simples no API Gateway \(p. 554\)](#)
- [Definição do OpenAPI da API de calculadora simples \(p. 558\)](#)

Gerar SDKs para uma API usando o console do API Gateway

Para gerar um SDK específico de linguagem ou de plataforma para uma API no API Gateway, você deve primeiro criar, testar e implantar a API em um estágio. Para fins de ilustração, usamos a API de [Calculadora simples \(p. 558\)](#) como exemplo para gerar SDKs específicos do linguagem ou plataforma nessa seção. Para obter instruções sobre como criar, testar e implantar essa API, consulte [Criar a API de Calculadora simples \(p. 554\)](#).

Tópicos

- [Gerar o SDK do Java de uma API \(p. 548\)](#)
- [Gerar o SDK do Android de uma API \(p. 549\)](#)
- [Gerar o SDK do iOS de uma API \(p. 549\)](#)
- [Gerar o SDK do JavaScript de uma API \(p. 550\)](#)
- [Gerar o SDK do Ruby de uma API \(p. 550\)](#)

Gerar o SDK do Java de uma API

Para gerar o SDK do Java de uma API no API Gateway

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Na caixa que contém o nome da API para o estágio, escolha Estágios.
3. No painel Stages (Estágios), escolha o nome do estágio.
4. Na guia SDK Generation, em Platform, escolha 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. Para Java Build System, digite `maven` ou `gradle` para especificar o sistema de build.
 - d. Para Java Group Id, digite um identificador de grupo para o seu projeto SDK. Por exemplo, `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. Para Java Artifact Id, digite um identificador de artefato para o seu projeto SDK. Por exemplo, `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, digite 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. Para Source Code License Text, digite o texto da licença do seu código fonte, se houver.
5. Escolha Generate SDK, e siga as instruções na tela para fazer download do SDK gerado pelo API Gateway.
 6. Siga as instruções em [Usar um SDK Java gerado pelo API Gateway para uma API REST \(p. 566\)](#) 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 do Android de uma API

Para gerar o SDK do Android de uma API no API Gateway

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Na caixa que contém o nome da API para o estágio, escolha Estágios.
3. No painel Stages (Estágios), escolha o nome do estágio.
4. Na guia SDK Generation, em Platform, escolha a plataforma Android.
 - a. Para Group ID, digite o identificador exclusivo para o projeto correspondente. Ele é usado no arquivo `pom.xml` (por exemplo, `com.mycompany`).
 - b. Para Invoker package, digite o namespace para as classes de cliente geradas (por exemplo, `com.mycompany.clientsdk`).
 - c. Para Artifact ID, digite 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, digite o número de versão do artefato para o cliente gerado. Ele é usado no arquivo `pom.xml` e deve seguir um padrão `principal.secundário.patch` (por exemplo, `1.0.0`).
5. Escolha Generate SDK, e siga as instruções na tela para fazer download do SDK gerado pelo API Gateway.
6. Siga as instruções em [Usar um SDK Android gerado pelo API Gateway para uma API REST \(p. 569\)](#) 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 do iOS de uma API

Para gerar o SDK do iOS de uma API no API Gateway

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Na caixa que contém o nome da API para o estágio, escolha Estágios.
3. No painel Stages (Estágios), escolha o nome do estágio.

4. Na guia SDK Generation, em Platform, escolha a plataforma iOS (Objective-C) or iOS (Swift).

- 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 \(p. 554\)](#) 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 [Use o SDK do iOS gerado pelo API Gateway para uma API REST em Objective-C ou Swift \(p. 575\)](#).

5. Escolha Generate SDK, e siga as instruções na tela para fazer download do SDK gerado pelo API Gateway.
6. Siga as instruções em [Use o SDK do iOS gerado pelo API Gateway para uma API REST em Objective-C ou Swift \(p. 575\)](#) 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 do JavaScript de uma API

Para gerar o SDK do JavaScript de uma API no API Gateway

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Na caixa que contém o nome da API para o estágio, escolha Estágios.
3. No painel Stages (Estágios), escolha o nome do estágio.
4. Na guia SDK Generation, em Platform, escolha JavaScript.
5. Escolha Generate SDK, e siga as instruções na tela para fazer download do SDK gerado pelo API Gateway.
6. Siga as instruções em [Usar um SDK JavaScript gerado pelo API Gateway para uma API REST \(p. 571\)](#) 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 do Ruby de uma API

Para gerar o SDK do Ruby de uma API no API Gateway

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Na caixa que contém o nome da API para o estágio, escolha Estágios.
3. No painel Stages (Estágios), escolha o nome do estágio.
4. Na guia SDK Generation, em Platform, escolha Ruby.
 - 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, este 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.

5. Escolha Generate SDK, e siga as instruções na tela para fazer download do SDK gerado pelo API Gateway.
6. Siga as instruções em [Usar um SDK Ruby gerado pelo API Gateway para uma API REST \(p. 573\)](#) 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

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 \(p. 551\)](#)
- [Gerar e fazer download do SDK do JavaScript usando a AWS CLI \(p. 551\)](#)
- [Gerar e fazer download do SDK do Ruby usando a AWS CLI \(p. 551\)](#)

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 (`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 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 um determinado estágio (`test`), use 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 [Chamar API REST por meio dos SDKs gerados \(p. 565\)](#).

Função Lambda da calculadora simples

Como ilustração, usaremos uma função Lambda Node.js que realiza as operações de binárias de adição, subtração, multiplicação e divisão.

Tópicos

- [Formato de entrada da função Lambda da calculadora simples \(p. 552\)](#)
- [Formato de saída da função Lambda da calculadora simples \(p. 552\)](#)
- [Implementação da função Lambda da calculadora simples \(p. 552\)](#)
- [Criar a função Lambda da calculadora simples \(p. 553\)](#)

Formato de entrada da função Lambda da calculadora simples

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 Lambda da calculadora simples

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 Lambda da calculadora simples

A implementação da função Lambda é a seguinte:

```
console.log('Loading the calculator function');

exports.handler = function(event, context, callback) {
    console.log('Received event:', JSON.stringify(event, null, 2));
    if (event.a === undefined || event.b === undefined || event.op === undefined) {
        callback("400 Invalid Input");
    }
}
```

```
var res = {};
res.a = Number(event.a);
res.b = Number(event.b);
res.op = event.op;

if (isNaN(event.a) || isNaN(event.b)) {
    callback("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":
        res.c = res.b==0 ? NaN : Number(event.a) / Number(event.b);
        break;
    default:
        callback("400 Invalid Operator");
        break;
}
callback(null, res);
};
```

Criar a função Lambda da calculadora simples

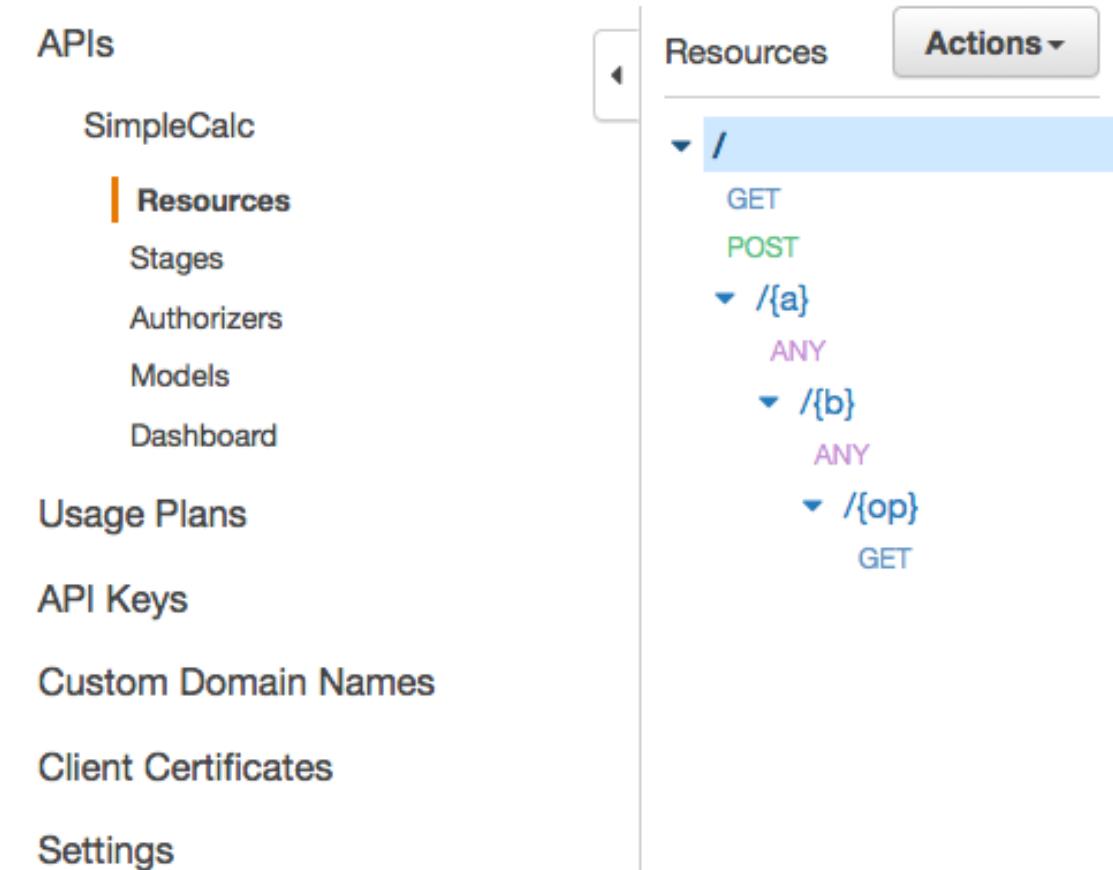
Você pode usar o console do AWS Lambda em <https://console.aws.amazon.com/lambda/> para criar a função, colando a listagem de código acima no editor de código online, da seguinte maneira.

The screenshot shows the AWS Lambda function configuration interface. At the top, there are tabs for Qualifiers, Test, and Actions. Below that, there are tabs for Code, Configuration, Triggers, and Monitoring. The Code tab is selected. In the main area, there is a dropdown for 'Code entry type' with 'Edit code inline' selected. The code editor contains the provided JavaScript code. Line 6 is highlighted with a light blue background, indicating it is the current line of interest.

```
1 console.log('Loading the Calc function');
2
3 exports.handler = function(event, context) {
4     console.log('Received event:', JSON.stringify(event, null, 2));
5     if (event.a === undefined || event.b === undefined || event.op === undefined) {
6         context.fail("400 Invalid Input");
7     }
8
9     var res = {};
10    res.a = Number(event.a);
11    res.b = Number(event.b);
12    res.op = event.op;
13}
```

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 Lambda da calculadora simples” \(p. 552\)](#). Uma representação gráfica dessa API é mostrada da seguinte forma:



Esses três métodos mostram maneiras diferentes de fornecer a entrada para a função do Lambda do 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 gera o nome do método de SDK correspondente, combinando o método de HTTP e as partes do caminho. 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 [Criação de uma API REST no Amazon API Gateway \(p. 182\)](#). Se você é novo com o API Gateway, consulte [Criar uma API do API Gateway com integração à Lambda \(p. 28\)](#) primeiro.

Criar modelos para entrada e saída

Para especificar uma entrada de tipo forte no SDK, criamos um modelo **Input** para a API:

Provide a name, content type, and a schema for your model. Models use [JSON schema](#).

Model name* Input

Content type* application/json

Model description

Model schema*

```

1 - {
2   "$schema": "http://json-schema.org/draft-04/schema#",
3   "type": "object",
4   "properties": {
5     "a": {"type": "number"},
6     "b": {"type": "number"},
7     "op": {"type": "string"}
8   },
9   "title": "Input"
10 }

```

* Required Cancel Create model

Da mesma forma, para descrever o tipo de dados do corpo de resposta, criamos os seguintes modelos no API Gateway:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "c": {"type": "number"}
  },
  "title": "Output"
}
```

e

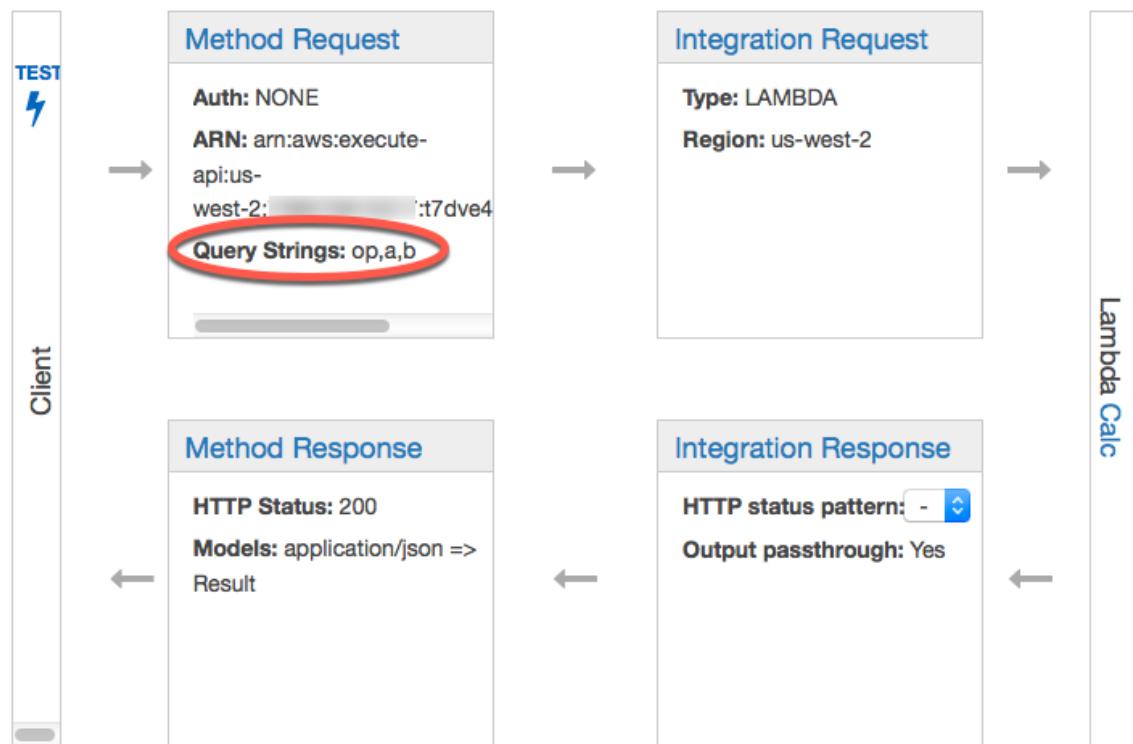
```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "input": {
      "$ref": "https://apigateway.amazonaws.com/restapis/t7dve4zn36/models/Input"
    },
    "output": {
      "$ref": "https://apigateway.amazonaws.com/restapis/t7dve4zn36/models/Output"
    }
},
```

```
        "title": "Result"  
    }
```

Configurar parâmetros da consulta do método GET /

Para o método GET /?a=..&b=..&op=.., os parâmetros de consulta estão declarados em Method Request:

/ - GET - Method Execution



Configurar o modelo de dados para a carga como entrada para o back-end

Para o método POST /, criamos o modelo Input e o adicionamos à solicitação de método para definir a forma dos dados de entrada.

[Method Execution](#) / - POST - Method Request

Provide information about this method's authorization settings and the parameters it can receive.

Authorization Settings

Authorization NONE 

API Key Required false 

▶ URL Query String Parameters

▶ HTTP Request Headers

▼ Request Models [Create a Model](#)

Content type	Model name	
application/json	Input 	

 [Add model](#)

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 Lambda.

[Configurar o modelo de dados para a saída do resultado do back-end](#)

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 Lambda.

[Method Execution](#) /{a}/{b}/{op} - GET - Method Response

Provide information about this method's response types, their headers and content types.

HTTP Status	
▼ 200 	

Response Headers for 200

Name	
No headers	
+ Add Header	

Response Models for 200

Create a model	
Content type	Models
application/json	 
	+ Add Response Model

[+ Add Response](#)

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.

Além disso, você também pode criar e configurar a API seguindo as [definições \(p. 558\)](#) de API do Swagger.

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 Lambda \(p. 552\)](#) após a importação. Para fazer isso, volte a selecionar a função Lambda criada na sua conta a partir da 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 do Lambda [add-permission](#).

OpenAPI 2.0

```
{
  "swagger": "2.0",
  "info": {
    "version": "2016-09-29T20:27:30Z",
    "title": "SimpleCalc"
  },
  "host": "t6dve4zn25.execute-api.us-west-2.amazonaws.com",
  "basePath": "/demo",
  "schemes": [
    "https"
  ],
  "paths": {
    "/": {
      "get": {
        "consumes": [

```

```
        "application/json"
    ],
    "produces": [
        "application/json"
    ],
    "parameters": [
        {
            "name": "op",
            "in": "query",
            "required": false,
            "type": "string"
        },
        {
            "name": "a",
            "in": "query",
            "required": false,
            "type": "string"
        },
        {
            "name": "b",
            "in": "query",
            "required": false,
            "type": "string"
        }
    ],
    "responses": {
        "200": {
            "description": "200 response",
            "schema": {
                "$ref": "#/definitions/Result"
            }
        }
    },
    "x-amazon-apigateway-integration": {
        "requestTemplates": {
            "application/json": "#set($inputRoot = $input.path('$'))\n\\\"a\\\" : $input.params('a'),\\n  \\\"b\\\" : $input.params('b'),\\n  \\\"op\\\" :\\n\\\"$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\\\"input\\\" : {\\n    \\\"a\\\" : $inputRoot.a,\\n    \\\"b\\\" : $inputRoot.b,\\n    \\\"op\\\" :\\n\\\"$inputRoot.op\\\"\n},\\n  \\\"output\\\" : {\\n    \\\"c\\\" : $inputRoot.c\\n  }\\n}"
                }
            },
            "type": "aws"
        }
    },
    "post": {
        "consumes": [
            "application/json"
        ],
        "produces": [
            "application/json"
        ],
        "parameters": [
            {
                "in": "body",
                "name": "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\\\"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\" : $input.params('b'),\n  \"op\" :\n\"$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\" : $inputRoot.b,\n    \"op\" :\n\"$inputRoot.op\"\n},\n    \"output\" : {\n    \"c\" : $inputRoot.c\n}\n"
                }
            }
        }
    }
},
"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"
      }
    }
  }
```

Como invocar uma API REST no Amazon API Gateway

Chamar uma API implantada envolve enviar solicitações à URL do serviço de componente do API Gateway para execução dessa API, conhecido como `execute-api`.

A URL base para APIs REST está no seguinte formato:

```
https://{restapi_id}.execute-api.{region}.amazonaws.com/{stage_name}/
```

em que `{restapi_id}` é o identificador da API, `{region}` é a região e `{stage_name}` é o nome do estágio da implantação de API.

Important

Antes de invocar uma API, você deve implantá-la no API Gateway. Para isso, siga as instruções em [Implantando uma API REST no Amazon API Gateway \(p. 516\)](#).

Tópicos

- [Obter a URL de invocação de uma API no console do API Gateway \(p. 563\)](#)
- [Usar o console do API Gateway para testar um método de API REST \(p. 564\)](#)
- [Usar o Postman para chamar uma API REST \(p. 565\)](#)
- [Chamar API REST por meio dos SDKs gerados \(p. 565\)](#)
- [Chamar a API REST por meio da biblioteca JavaScript AWS Amplify \(p. 584\)](#)
- [Como invocar uma API privada \(p. 584\)](#)

Obter a URL de invocação de uma API no console do API Gateway

Você pode encontrar uma URL raiz da API REST no Editor de estágio para a API no console do API Gateway. Ela está listado como Invoke URL (Invocar URL) na parte superior. Se o recurso raiz da API

expuser um método `GET` sem exigir a autenticação do usuário, você poderá chamar o método clicando no link `Invoke URL` (`Invocar URL`). Também é possível construir esse URL raiz combinando os campos `host` e `basePath` de um arquivo de definição do OpenAPI exportado da API.

Se uma API permitir acesso anônimo, você poderá usar qualquer navegador da Web para invocar qualquer chamada de método `GET`, copiando e colando um URL de invocação apropriado na barra de endereço do navegador. Para outros métodos ou qualquer chamada exigida para autenticação, a invocação será mais envolvida, pois você deve especificar uma carga ou assinar as solicitações. É possível lidar com elas em um script atrás de uma página HTML ou em um aplicativo cliente usando um dos SDKs da AWS.

Para testes, você pode usar o console do API Gateway para chamar uma API usando o recurso `TestInvoke` do API Gateway, que ignora a URL `Invoke` e permite testes da API antes que ela seja implantada. Como alternativa, você pode usar o aplicativo [Postman](#) para testar uma API implantada com êxito, sem escrever um script ou um cliente.

Note

Os valores de parâmetros de strings de consulta em uma URL de invocação não podem conter %.

Usar o console do API Gateway para testar um método de API REST

Usar o console do API Gateway para testar um método de API REST.

Tópicos

- [Pré-requisitos \(p. 564\)](#)
- [Testar um método com o console do API Gateway \(p. 564\)](#)

Pré-requisitos

- Você deve especificar as configurações dos métodos que deseja testar. Siga as instruções em [Configurar métodos de API REST no API Gateway \(p. 204\)](#).

Testar um método com o console do API Gateway

Important

Testar métodos com o console do API Gateway pode resultar em alterações em recursos que não podem ser desfeitas. Testar um método com o console do API Gateway é o mesmo que chamar esse 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.

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Na caixa que contém o nome da API para o método, escolha Recursos.
3. No painel Resources (Recursos), escolha o método que você deseja testar.
4. No painel Method Execution (Execução de método), na caixa Client (Cliente), escolha TEST. Digite os valores em qualquer das caixas exibidas (tais como Query Strings (Strings de consulta), Headers (Cabeçalhos) e Request Body (Corpo da solicitação)). O console inclui esses valores na solicitação de método no formulário json/aplicativo padrão.

Para opções adicionais você talvez precise especificar, entre em contato com o proprietário da API.

5. Escolha Test. 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.
- Latency (Latência) é o tempo entre a recepção da solicitação do chamador e a resposta retornada.
- Response Body (Corpo da resposta) é o corpo de resposta HTTP.
- Response Headers (Cabeçalhos de resposta) são os cabeçalhos de resposta HTTP.

Tip

Dependendo do mapeamento, o código de status HTTP, o corpo de resposta e os cabeçalhos de resposta podem ser diferentes dos enviados pela função do Lambda, pelo proxy HTTP ou pelo proxy de serviço da AWS.

- 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 SDK da AWS para o API Gateway a fim de testar a invocação de um método. Para fazer isso usando a AWS CLI, consulte [test-invoke-method](#).

Usar o Postman para chamar uma API REST

O aplicativo [Postman](#) é uma ferramenta prática para testar uma API REST no API Gateway. As instruções a seguir descreverão as etapas essenciais de uso do aplicativo Postman para chamar uma API. Para obter mais informações, consulte a [ajuda](#) do Postman.

1. Inicie o Postman.
2. Insira a URL de endpoint de uma solicitação na barra de endereço e escolha o método HTTP apropriado na lista suspensa à esquerda da barra de endereço.
3. Se necessário, escolha a guia Authorization (Autorização). Escolha AWS Signature (Assinatura da AWS) para a autorização Type (Tipo). Digite seu ID de chave de acesso de usuário AWS ao IAM no campo de entrada AccessKey. Insira a chave secreta do usuário IAM em SecretKey. Especifique uma região da AWS apropriada que corresponda à região especificada na URL de invocação. Insira **execute-api** em Service Name (Nome do serviço).
4. Escolha a guia Headers. Opcionalmente, exclua todos os cabeçalhos existentes. Isso pode limpar qualquer configuração obsoleta que possa causar erros. Adicione todos os cabeçalhos personalizados necessários. Por exemplo, se as chaves de API estiverem habilitadas, você poderá definir o par de nome/valor **x-api-key:{api_key}** aqui.
5. Escolha Send para enviar a solicitação e receber uma resposta.

Para ver um exemplo de como usar o Postman, consulte [Chamar uma API com autorizadores do Lambda do API Gateway \(p. 409\)](#).

Chamar API REST por meio dos SDKs gerados

Esta seção mostra como chamar uma API por meio de um SDK em um aplicativo cliente programado em Java, Java para Android, JavaScript, Ruby, Objective-C e Swift.

Tópicos

- Usar um SDK Java gerado pelo API Gateway para uma API REST (p. 566)
- Usar um SDK Android gerado pelo API Gateway para uma API REST (p. 569)
- Usar um SDK JavaScript gerado pelo API Gateway para uma API REST (p. 571)
- Usar um SDK Ruby gerado pelo API Gateway para uma API REST (p. 573)
- Use o SDK do iOS gerado pelo API Gateway para uma API REST em Objective-C ou Swift (p. 575)

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 \(p. 558\)](#) como exemplo. Antes de prosseguir, você deve concluir as etapas em [Gerar SDKs para uma API usando o console do API Gateway \(p. 548\)](#).

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.awslambda.simpleCalc.sdk.*;
import examples.awslambda.simpleCalc.sdk.model.*;
import examples.awslambda.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.geteResultByPostInputBody(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 uma API usando o console do API Gateway \(p. 548\)](#).

Note

O SDK gerado não é compatível com o Android 4.4 e versões anteriores. Para obter mais informações, consulte [the section called “Observações importantes” \(p. 730\)](#).

Para instalar e usar um SDK do 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 SDK (cujo nome é derivado do Artifact Id e do Artifact Version que você especificou ao gerar o SDK, por exemplo, `simple-calcsdk-1.0.0.jar`) na 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

Note

Estas instruções supõem que você já concluiu as instruções em [Gerar SDKs para uma API usando o console do API Gateway \(p. 548\)](#).

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 [Habilitar o CORS para um recurso da API REST \(p. 423\)](#).
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. aBOpGet(params);      // where params={"a": ..., "b": ..., "op": ...} is resolved to  
the path parameters
```

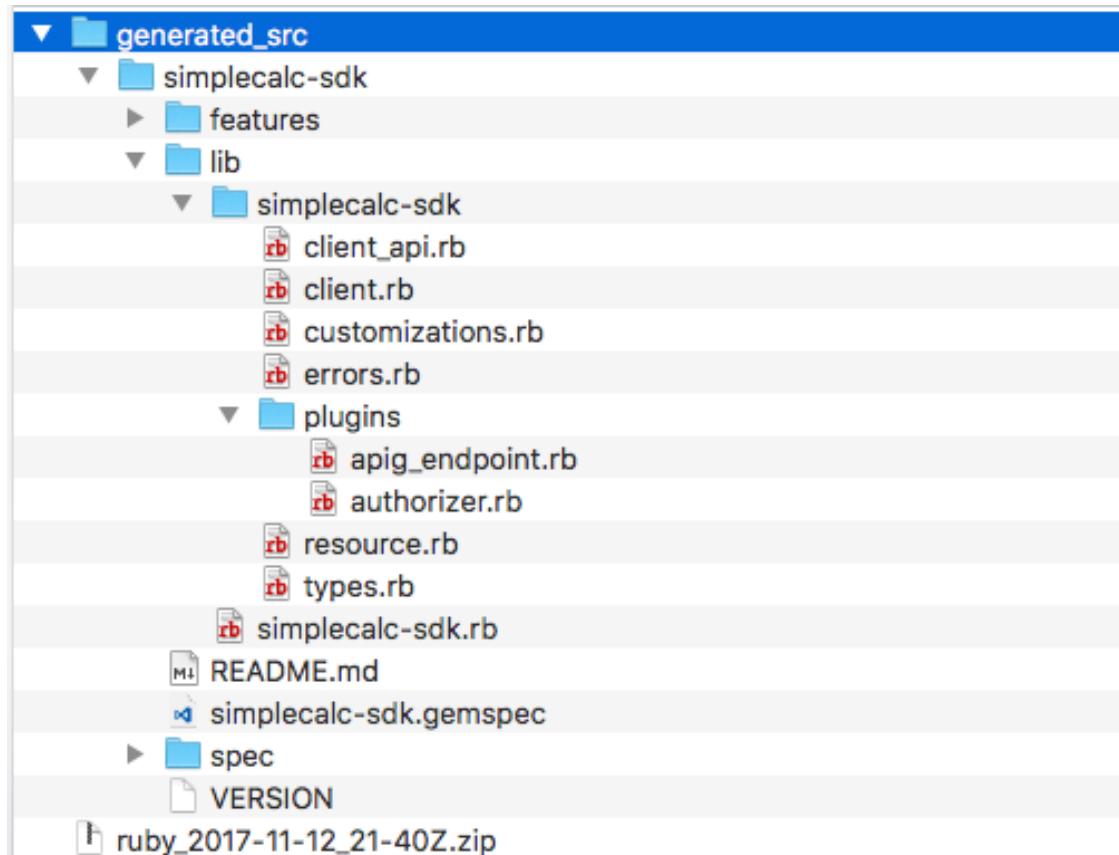
Usar um SDK Ruby gerado pelo API Gateway para uma API REST

Note

Estas instruções supõem que você já tenha concluído as instruções em [Gerar SDKs para uma API usando o console do API Gateway \(p. 548\)](#).

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
```

- 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 chamador, 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: 'ACCESS_KEY',  
    secret_key: 'SECRET_KEY'  
)
```

- 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
        }
    }
}
```

Use o SDK do 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 \(p. 554\)](#) 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 \(p. 575\)](#)
- [Usar o SDK do iOS \(Swift\) gerado para chamar a API \(p. 579\)](#)

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 uma API usando o console do API Gateway \(p. 548\)](#) para o iOS no Objective-C e fazer download do 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

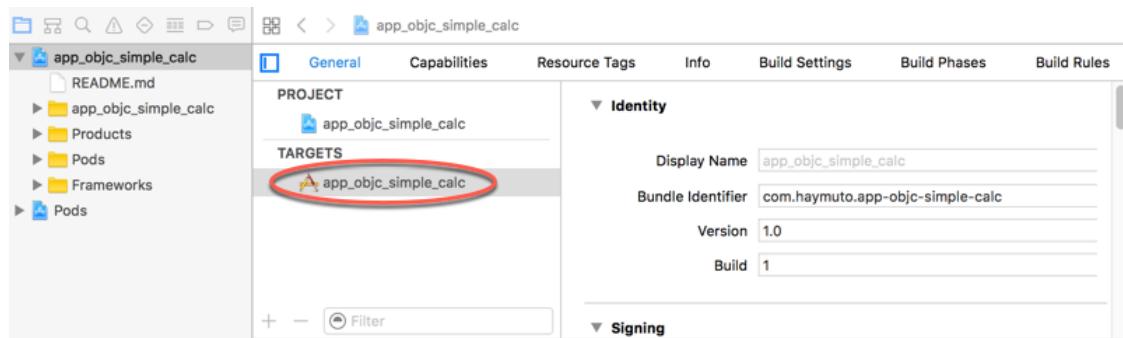
O procedimento a seguir descreve como instalar o SDK.

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 \(p. 554\)](#), 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` 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:

- Instale o CocoaPods executando o seguinte comando em uma janela de terminal:

```
sudo gem install cocoapods  
pod setup
```

- 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'
```

- 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.

- Feche o projeto Xcode e abra o arquivo `.xcworkspace` para reiniciar o Xcode.
- Adicione todos os arquivos `.h` e `.m` do diretório `generated-src` do SDK extraído ao seu projeto Xcode.

```

1 /*
2 Copyright 2010-2016 Amazon.com, Inc. or its affiliates. All Rights
3 Reserved.
4
5 Licensed under the Apache License, Version 2.0 (the "License").
6 You may not use this file except in compliance with the License.
7 A copy of the License is located at
8
9 http://aws.amazon.com/apache2.0
10
11 or in the "license" file accompanying this file. This file is distributed
12 on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
13 express or implied. See the License for the specific language governing
14 permissions and limitations under the License.
15 */
16
17 #import "SIMPLE_CALCSimpleCalcClient.h"
18 #import <AWSCore/AWSCore.h>
19 #import <AWSCore/AWSSignature.h>
20 #import <AWSCore/AWSynchronizedMutableDictionary.h>
21
22 #import "SIMPLE_CALCResult.h"
23 #import "SIMPLE_CALCInput.h"
24
25 @interface AWSAPIGatewayClient()
26
27 // Networking
28 @property (nonatomic, strong) NSURLSession *session;
29
30 // For requests
31 @property (nonatomic, strong) NSURL *baseURL;
32
33 // For responses
34 @property (nonatomic, strong) NSDictionary *HTTPHeaderFields;
35 @property (nonatomic, assign) NSInteger HTTPStatusCode;
36
37 - (AWSTask *)invokeHTTPRequest:(NSString *)HTTPMethod

```

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](#) (p. 554) 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.

Para 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:@"%@ %@ %@ = %@", 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 aBOpGet:@"1" b:@"2" op:@"div"] continueWithBlock:^id _Nullable(AWSTask * _Nonnull task) {
    _textField3.text = [self handleApiResponse: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 \(p. 552\)](#) no back-end 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 uma API usando o console do API Gateway \(p. 548\)](#) para o iOS no Swift e fazer download do arquivo .zip do SDK gerado.

Tópicos

- [Instalar o SDK Móvel da AWS e o SDK gerado pelo API Gateway em um projeto Swift \(p. 579\)](#)
- [Chamar métodos de API por meio do SDK do iOS gerado pelo API Gateway em um projeto Swift \(p. 582\)](#)

Instalar o SDK Móvel da AWS e o SDK gerado pelo API Gateway em um projeto Swift

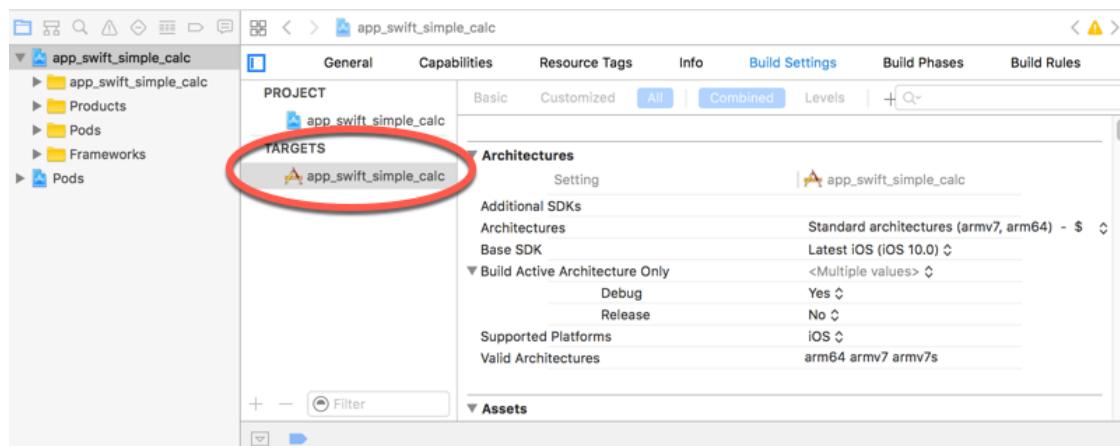
O procedimento a seguir descreve como instalar o SDK.

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 \(p. 554\)](#), 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'
```

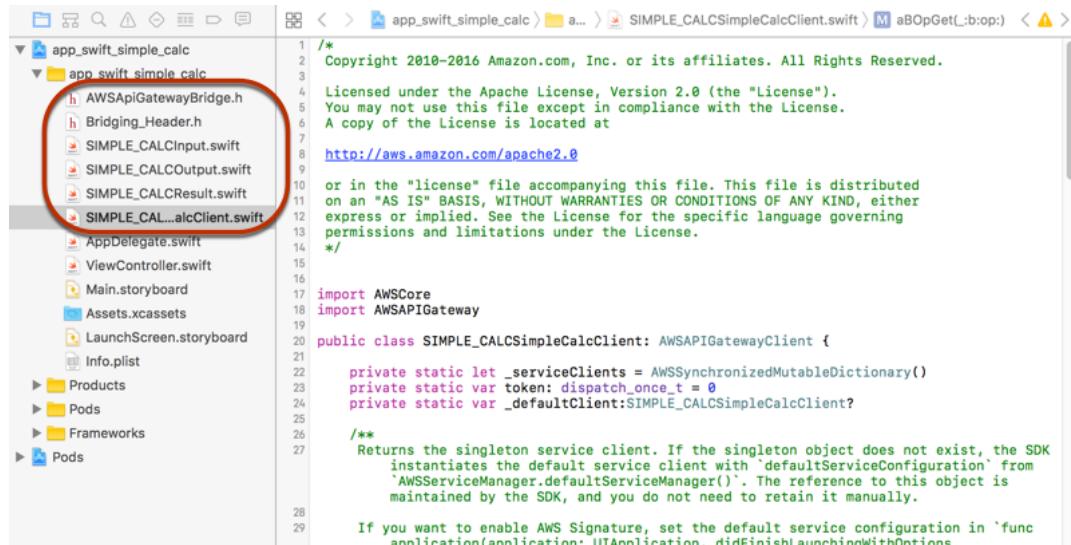
- c. Abra uma janela de terminal e execute o seguinte comando no diretório do aplicativo:

```
pod install
```

Isso instala o componente do API Gateway e quaisquer componentes dependentes do SDK Móvel da AWS no projeto do aplicativo.

- 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 generated-src para seu projeto Xcode.



The screenshot shows the Xcode project structure for 'app_swift_simple_calc'. The 'generated-src' folder contains several files, with 'SIMPLE_CALCSimpleCalcClient.swift' being the most prominent. This file includes imports for AWSCore and AWSSAPIGateway, and defines a class 'SIMPLE_CALCSimpleCalcClient' that inherits from 'AWSSAPIGatewayClient'. A specific line of code is highlighted: 'private static var _defaultClient:SIMPLE_CALCSimpleCalcClient?'. The entire file is annotated with comments explaining its purpose and usage.

```

/*
Copyright 2010-2016 Amazon.com, Inc. or its affiliates. All Rights Reserved.

Licensed under the Apache License, Version 2.0 (the "License").
You may not use this file except in compliance with the License.
A copy of the License is located at

http://aws.amazon.com/apache2.0

or in the "license" file accompanying this file. This file is distributed
on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
express or implied. See the License for the specific language governing
permissions and limitations under the License.

*/
import AWSCore
import AWSSAPIGateway

public class SIMPLE_CALCSimpleCalcClient: AWSSAPIGatewayClient {

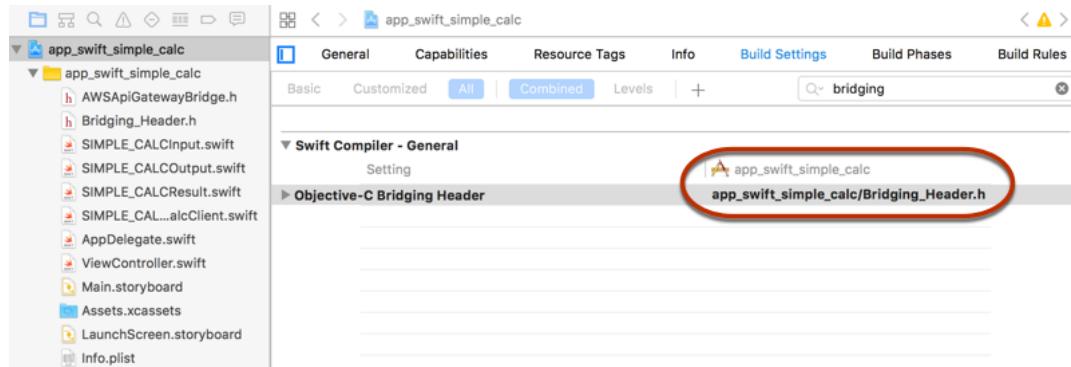
    private static let _serviceClients = AWSSynchronizedMutableDictionary()
    private static var token: dispatch_once_t = 0
    private static var _defaultClient:SIMPLE_CALCSimpleCalcClient?

    /**
     Returns the singleton service client. If the singleton object does not exist, the SDK
     instantiates the default service client with 'defaultServiceConfiguration' from
     'AWSServiceManager.defaultServiceManager()'. The reference to this object is
     maintained by the SDK, and you do not need to retain it manually.

     If you want to enable AWS Signature, set the default service configuration in 'func
     application(application: UIApplication, didFinishLaunchingWithOptions'

```

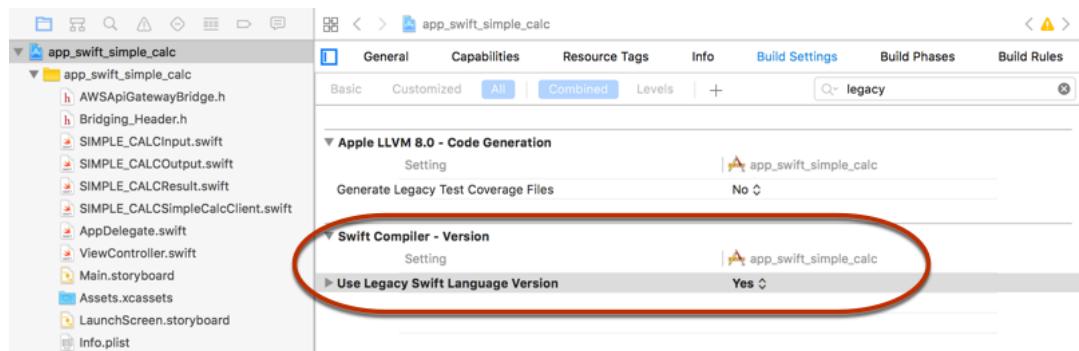
- f. Para permitir a chamada de bibliotecas Objective-C do SDK Móvel da AWS a partir do seu projeto de código Swift, defina o caminho do arquivo Bridging_Header.h na propriedade Objective-C Bridging Header (Cabeçalho ponte Objective-C), na definição Swift Compiler - General (Compilador Swift - Geral) 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 SDK móvel da AWS para iOS em Swift no seu projeto, baixando explicitamente esse SDK da AWS ou usando o [Carthage](#), siga as instruções no arquivo README .md que acompanha o pacote 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 \(p. 554\)](#) 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 aBOpGet(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`.

Para 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.defaultClient()
```

Para usar o Amazon Cognito com a API, defina uma configuração de serviço da AWS padrão (mostrada a seguir) antes de obter o método `defaultClient` (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 \(p. 552\)](#) no back-end não manuseia variáveis de caminho codificadas por URL.

Chamar a API REST por meio da biblioteca JavaScript AWS Amplify

A biblioteca JavaScript AWS Amplify pode ser usada para fazer solicitações de API a uma API REST do API Gateway. Para obter mais informações, consulte as instruções no [Guia de API do AWS Amplify](#).

Como invocar uma API privada

As APIs privadas podem ser acessadas somente em suas VPCs, e as [políticas de recursos \(p. 204\)](#) devem permitir o acesso das VPCs e dos VPC endpoints que você configurou. O modo como você acessa sua API privada dependerá de você ter habilitado ou não o DNS privado no VPC endpoint. Por exemplo, ao acessar a API privada da rede local por meio do AWS Direct Connect, o DNS privado será habilitado no VPC endpoint. Nesse caso, siga as etapas descritas em [Invocar sua API privada usando nomes de host DNS públicos específicos do endpoint \(p. 586\)](#).

Depois que você tiver implantado uma [API privada \(p. 200\)](#), poderá acessá-la por meio do DNS privado (se a nomeação de DNS privado estiver habilitada) e do DNS público.

Para obter os Nomes de DNS para sua API privada, siga estas instruções:

1. Faça login no console da Amazon VPC em <https://console.aws.amazon.com/vpc/>.
2. No painel de navegação à esquerda, selecione Endpoints e escolha o VPC endpoint de interface do API Gateway.
3. No painel Details (Detalhes), você verá cinco valores no campo DNS names (Nomes DNS). Os três primeiros são os nomes DNS público para a sua API. Os outros dois são os nomes DNS privado para ela.

Invocar a API privada usando os Nomes de DNS privado

Warning

Quando você seleciona a opção Enable Private DNS Name (Habilitar nome DNS privado) ao criar um VPC endpoint de interface para o API Gateway, a VPC onde o VPC Endpoint está não poderá acessar APIs públicas (otimizadas para fronteiras e regionais). Para obter mais informações, consulte [Por que não consigo me conectar à API pública em um VPC endpoint do API Gateway?](#).

Se você tiver habilitado o DNS privado, poderá acessar sua API privada usando os Nomes de DNS privado da seguinte forma:

```
{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, supondo que você configure os métodos GET /pets e GET /pets/{petId} neste exemplo e supondo que o ID da API REST fosse 01234567ab e sua região fosse us-west-2, seria possível testar a API digitando os seguintes URLs em um navegador:

```
https://01234567ab.execute-api.us-west-2.amazonaws.com/test/pets
```

e

```
https://01234567ab.execute-api.us-west-2.amazonaws.com/test/pets/1
```

Se desejar, você pode usar os seguintes comandos cURL:

```
curl -X GET https://01234567ab.execute-api.us-west-2.amazonaws.com/test/pets
```

e

```
curl -X GET https://01234567ab.execute-api.us-west-2.amazonaws.com/test/pets/2
```

Acessar a API privada usando o AWS Direct Connect

Você também pode usar o AWS Direct Connect para estabelecer uma conexão privada dedicada de uma rede local para a Amazon VPC e acessar seu endpoint da API privada nessa conexão usando nomes DNS público.

Não é possível usar Nomes de DNS privado para acessar a API privada a partir de uma rede local.

Acessar sua API privada usando um alias do Route53

É possível associar ou desassociar um VPC endpoint de sua API privada usando o procedimento descrito em [Associar ou desassociar um VPC endpoint de uma API REST privada \(p. 515\)](#).

Depois de associar o ID da API REST da sua API privada aos VPC endpoints dos quais você chamará a API REST, será possível usar o URL base no formato a seguir para invocar a API usando um alias do Route53.

O URL base gerado está no seguinte formato:

```
https://{rest-api-id}-{vpce-id}.execute-api.{region}.amazonaws.com/{stage}
```

Por exemplo, supondo que você configure os métodos GET/pets e GET/pets/{petId} neste exemplo, e supondo que o ID de sua API seja 01234567ab, o ID do VPC Endpoint seja vpce-01234567abcdef012 e sua região seja us-west-2, é possível invocar sua API como:

```
curl -v https://01234567ab-vpce-01234567abcdef012.execute-api.us-west-2.amazonaws.com/test/pets/
```

Invocar a API privada usando nomes de host DNS públicos específicos de 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, supondo que você configure os métodos GET /pets e GET /pets/{petId} neste exemplo, e supondo que o ID de sua API seja 01234567ab e o nome do host DNS público seja vpce-01234567abcdef012-01234567 e sua região seja us-west-2, é possível testar sua API pelo ID de VPCE usando o cabeçalho Host em um comando cURL, conforme mostrado no seguinte exemplo:

```
curl -v https://vpce-01234567abcdef012-01234567.execute-api.us-east-1.vpce.amazonaws.com/test/pets -H 'Host: 01234567ab.execute-api.us-west-2.amazonaws.com'
```

Se preferir, acesse a API privada pelo ID da 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/test -H'x-apigw-api-id:{api-id}'
```

Rastreamento, registro em logs e monitoramento de uma API do API Gateway

[AWS X-Ray](#), [AWS CloudTrail](#) e [Amazon CloudWatch](#) são ferramentas que os desenvolvedores do Amazon API Gateway podem usar para rastrear, registrar em log e monitorar a execução e operações de gerenciamento da API.

AWS X-Ray é um serviço da AWS que permite rastrear problemas de latência com as APIs do Amazon API Gateway. O X-Ray coleta metadados do serviço API Gateway e quaisquer serviços downstream que compõem sua API. O X-Ray usa esses metadados para gerar um gráfico de serviço detalhado que ilustra os picos de latência e outros problemas que afetam o desempenho da API.

O Amazon CloudWatch registra em log as operações de execução da API, que são chamadas que um cliente ou aplicativo cliente da API faz contra um componente execute-api do API Gateway. As métricas do CloudWatch incluem estatísticas sobre o armazenamento em cache, a latência e os erros detectados. Você pode inspecionar os logs do CloudWatch para solucionar problemas em sua implementação ou execução da API usando o painel da API no console do API Gateway ou usando o console do CloudWatch.

O AWS CloudTrail registra em log as operações de gerenciamento da API do API Gateway, que são chamadas da API REST que um desenvolvedor ou proprietário da API faz contra o componentes apigateway do API Gateway. Os logs do CloudTrail podem ser usados para solucionar problemas com a criação, a implantação e as atualizações de APIs. Você também pode usar o Amazon CloudWatch para monitorar logs do CloudTrail.

Tópicos

- [Rastrear execução da API do API Gateway com o AWS X-Ray \(p. 587\)](#)

- [Registrar chamadas para APIs do Amazon API Gateway com AWS CloudTrail \(p. 597\)](#)
- [Monitorar a execução da API com o Amazon CloudWatch \(p. 599\)](#)
- [Usar o Amazon Kinesis Data Firehose como destino para registro de acesso em logs do API Gateway \(p. 605\)](#)

Rastrear execução da API do API Gateway com o AWS X-Ray

Você pode usar o [AWS X-Ray](#) para rastrear e analisar as solicitações do usuário à medida que são transferidas por meio das APIs do Amazon API Gateway para os serviços subjacentes. O API Gateway é compatível com o rastreamento do AWS X-Ray para todos os tipos de endpoint do API Gateway: regional, otimizado para fronteiras e privados. Use o AWS X-Ray com o Amazon API Gateway em todas as regiões onde o X-Ray estiver disponível.

O X-Ray fornece uma visão completa de toda uma solicitação, para que você possa analisar latências nas APIs e seus serviços de back-end. Use um mapa de serviço do X-Ray para visualizar a latência de toda uma solicitação e dos serviços downstream integrados ao X-Ray. E você pode configurar regras de amostragem para informar ao X-Ray quais solicitações registrar, com quais taxas de amostragem, de acordo com os critérios especificados. Se você chamar uma API do API Gateway a partir de um serviço que já foi rastreado, o API Gateway envia o rastreamento adiante, 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 de gerenciamento do API Gateway ou usando a API ou a CLI do API Gateway.

Tópicos

- [Como configurar AWS X-Ray com o API Gateway \(p. 587\)](#)
- [Uso de mapas de serviço e exibições de rastreamento do AWS X-Ray com o API Gateway \(p. 590\)](#)
- [Configuração de regras de amostragem do AWS X-Ray para APIs do API Gateway \(p. 593\)](#)
- [Noções básicas sobre os rastreamentos do AWS X-Ray para APIs do Amazon API Gateway \(p. 595\)](#)

Como configurar AWS X-Ray com o API Gateway

Nesta seção você pode encontrar informações detalhadas sobre como configurar o [AWS X-Ray](#) com o API Gateway.

Tópicos

- [Modelos de rastreamento do X-Ray para o API Gateway \(p. 587\)](#)
- [Permissões para o rastreamento do X-Ray \(p. 588\)](#)
- [Habilitação do rastreamento do X-Ray no console do API Gateway \(p. 588\)](#)
- [Habilitar o rastreamento do AWS X-Ray usando a CLI do API Gateway \(p. 588\)](#)

Modelos 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 é a configuração padrão caso não tenha habilitado o rastreamento do X-Ray em um estágio da API. Esta abordagem significa que a API do API Gateway só é rastreada se o X-Ray foi 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 a um serviço na sua conta, se a função ainda não existir. 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 é 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 o AWS SDK.

Permissões para o rastreamento do X-Ray

Ao habilitar o rastreamento do X-Ray em um estágio, o API Gateway cria uma função vinculada a um serviço na sua conta, se a função ainda não existir. 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](#).

Habilitação do 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. No painel APIs, escolha a API e, em seguida, Stages (Estágios).
3. No painel Stages (Estágios), escolha o nome do estágio.
4. No painel Stage Editor (Editor de estágio), escolha a guia Logs/Tracing (Registros/Rastreamento).
5. Para habilitar o rastreamento ativo do X-Ray, selecione Enable X-Ray Tracing (Habilitar o rastreamento do X-Ray) em X-Ray Tracing (Rastreamento do X-Ray).
6. Se quiser, escolha Set X-Ray Sampling Rules (Definir regras de amostragem do X-Ray) e acesse o console do X-Ray para [configurar regras de amostragem \(p. 593\)](#).

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

Para usar a AWS CLI para habilitar a rastreamento ativo do X-Ray para um estágio da API ao criar o estágio, chame o comando `create-stage` conforme o exemplo a seguir:

```
aws apigateway --endpoint-url {endpoint-url} create-stage
```

\

```
--rest-api-id {rest-api-id}      \
--stage-name {stage-name}        \
--deployment-id {deployment-id} \
--region {region}              \
--tracing-enabled=true
```

Veja a seguir um exemplo de saída para uma invocação bem-sucedida:

```
{
  "tracingEnabled": true,
  "stageName": {stage-name},
  "cacheClusterEnabled": false,
  "cacheClusterStatus": "NOT_AVAILABLE",
  "deploymentId": {deployment-id},
  "lastUpdatedDate": 1533849811,
  "createdDate": 1533849811,
  "methodSettings": {}
}
```

Para usar a AWS CLI para desabilitar o rastreamento ativo do X-Ray para um estágio da API ao criar o estágio, chame o comando [create-stage](#) conforme o exemplo a seguir:

```
aws apigateway --endpoint-url {endpoint-url} create-stage           \
--rest-api-id {rest-api-id}      \
--stage-name {stage-name}        \
--deployment-id {deployment-id} \
--region {region}              \
--tracing-enabled=false
```

Veja a seguir um exemplo de saída para uma invocação bem-sucedida:

```
{
  "tracingEnabled": false,
  "stageName": {stage-name},
  "cacheClusterEnabled": false,
  "cacheClusterStatus": "NOT_AVAILABLE",
  "deploymentId": {deployment-id},
  "lastUpdatedDate": 1533849811,
  "createdDate": 1533849811,
  "methodSettings": {}
}
```

Para usar a AWS CLI para habilitar o rastreamento ativo do X-Ray para uma API que já foi implantada, chame o comando [update-stage](#) da seguinte forma:

```
aws apigateway update-stage           \
--rest-api-id {rest-api-id}      \
--stage-name {stage-name}        \
--patch-operations op=replace,path=/tracingEnabled,value=true
```

Para usar a AWS CLI para desabilitar o rastreamento ativo do X-Ray para uma API que já foi implantada, chame o comando [update-stage](#) conforme o exemplo a seguir:

```
aws apigateway update-stage           \
--endpoint-url {endpoint-url}        \
--rest-api-id {rest-api-id}      \
--stage-name {stage-name}        \
--region {region}              \
```

```
--patch-operations op=replace,path=/tracingEnabled,value=false
```

Veja a seguir um exemplo de saída para uma invocação bem-sucedida:

```
{  
    "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. Para obter mais informações, consulte [Usar a API do AWS X-Ray com a CLI da AWS](#).

Uso de mapas de serviço e exibições 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.

Para obter informações detalhadas sobre mapas de serviço e exibições de rastreamento, e como interpretá-los, consulte o [Console do AWS X-Ray](#).

Tópicos

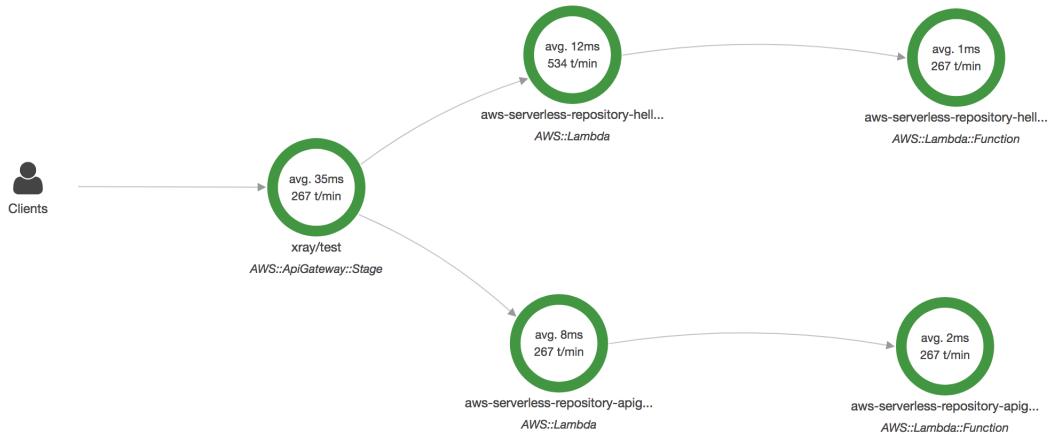
- [Exemplo de mapa de serviço do X-Ray \(p. 590\)](#)
- [Exemplo de exibição de rastreamento do X-Ray \(p. 593\)](#)

Exemplo de mapa de serviço do X-Ray

Os mapas de serviço do AWS X-Ray 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 do 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 integradas a serviços da AWS, como o AWS Lambda e o Amazon DynamoDB, você verá mais nós que fornecem métricas de desempenho 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`. A API possui uma integração ao Lambda com uma função de autorizador do Lambda e uma função de back-end do Lambda. Os nós representam o serviço do API Gateway, o serviço do Lambda e as duas funções do Lambda.

Para obter uma explicação detalhada da estrutura do mapa de serviço, consulte [Exibição do mapa de serviço](#).



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 do Lambda e a função do Lambda. Para obter mais informações, consulte [Lambda como um rastreamento do AWS X-Ray](#).

Se você escolher um nó ou um ponto em um mapa de serviço do X-Ray, o console do X-Ray mostra 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. Para obter uma explicação detalhada sobre os histogramas de distribuição de latência, consulte [Uso de histogramas de latência no console do AWS X-Ray](#).

Service details ?

Name: xray/test
Type: AWS::ApiGateway::Stage

Response distribution

Click and drag to select an area to zoom in on or use as a latency filter when viewing traces.

Duration (ms)	Percentage (%)
20	1.0
25	8.5
28	9.5
30	9.0
35	3.0
40	1.5
45	2.0
50	0.5
55	0.5
60	0.5

Response status

Choose response statuses to add to the filter when viewing traces.

OK: 100% Error: 0%

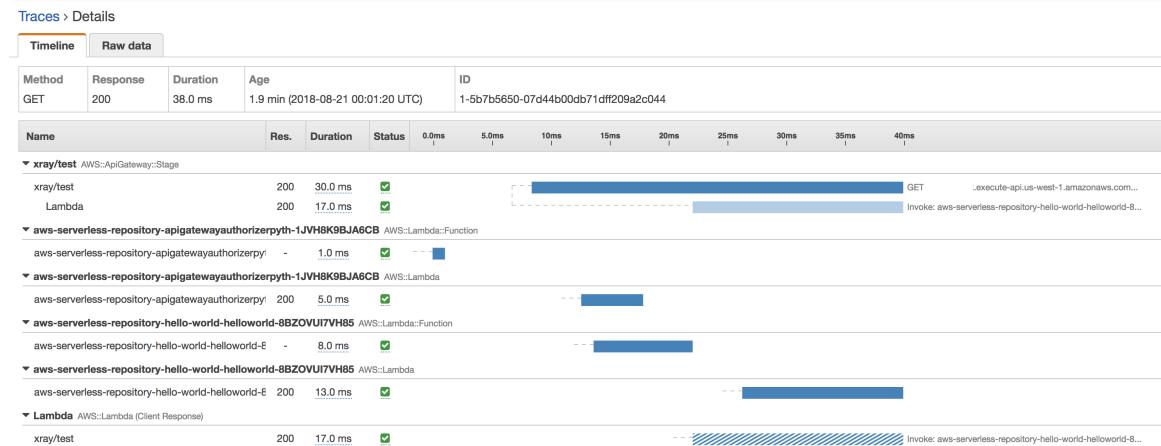
Fault: 0% Throttle: 0%

592 Close View traces >

Exemplo de exibição de rastreamento do X-Ray

O diagrama a seguir mostra uma exibição de rastreamento gerada para a API de exemplo descrita acima, com uma função de back-end do Lambda e uma função de autorizador do Lambda. Uma solicitação de método da API bem-sucedida é mostrada com um código de resposta de 200.

Para obter uma explicação detalhada sobre exibições de rastreamento, consulte [Visualização de rastreamentos](#).



Configuração de 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:

- [Configuração de regras de amostragem no console do AWS X-Ray](#)
- [Uso de 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 \(p. 593\)](#)
- [Exemplos de regras de amostragem do X-Ray \(p. 594\)](#)

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 (*). Para obter mais detalhes, incluindo uma explicação detalhada sobre como as configurações de Reservatório e Taxa são usadas, consulte [Configuração de regras de amostragem no console do AWS X-Ray](#).

- Nome da regra (string) — um nome exclusivo para a regra.

- Prioridade (íntero 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 (íntero 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 (p. 45) 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, no formato `arn:aws:execute-api:{region}:{account-id}:{api-id}/{stage-name}`, por exemplo, `arn:aws:execute-api:us-east-1:123456789012:qsxrty/test`.

O nome de estágio pode ser obtido a partir do console, da CLI ou da API do API Gateway. Para obter mais informações sobre os formatos de ARN, consulte a [Referência geral do Amazon Web Services](#).

- Método HTTP (string) — o método a ser amostrado, por exemplo, `GET`.
- Caminho do URL (string) — esta opção não é compatível com o API Gateway.
- (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.

- Nome da regra — `test-sampling`
- Prioridade — `17`
- Tamanho do reservatório — `10`
- Taxa fixa — `10`
- Nome de 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.

- Nome da regra — `prod-sampling`
- Prioridade — `478`
- Tamanho do reservatório — `1`

- Taxa fixa — **60**
- Nome de serviço — **testxray/prod**
- Tipo de serviço — **AWS::ApiGateway::Stage**
- Método HTTP — *
- ARN do recurso — *
- Host — *
- Atributos — {}

Noções básicas sobre os rastreamentos do AWS X-Ray para APIs do Amazon API Gateway

Esta seção discute segmentos, subsegmentos e outros campos de rastreamento do AWS X-Ray para APIs do Amazon API Gateway.

Antes de ler esta seção, reveja os tópicos a seguir no Guia do desenvolvedor do X-Ray:

- [Console do AWS X-Ray](#)
- [Documentos de segmento do AWS X-Ray](#)
- [Conceitos do X-Ray](#)

Tópicos

- [Exemplos de objetos de rastreamento para uma API do API Gateway \(p. 595\)](#)
- [Noções básicas sobre o rastreamento \(p. 596\)](#)

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. Para obter mais informações, consulte [Configuração das regras de amostragem no console do AWS X-Ray](#).

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"  
}
```

Dados de recursos da AWS

O objeto `aws` aparece somente em segmentos. Veja a seguir um exemplo de um objeto `aws` que corresponde à regra de amostragem Padrão. Para obter uma explicação detalhada sobre as regras de amostragem, consulte [Configuração das regras de amostragem no console do AWS X-Ray](#).

```
"aws": {
```

```
        "xray": {
            "sampling_rule_name": "Default"
        },
        "api_gateway": {
            "account_id": "123412341234",
            "rest_api_id": "a1b2c3d4e5",
            "stage": "dev",
            "request_id": "a1b2c3d4-a1b2-a1b2-a1b2-a1b2c3d4e5f6"
        }
    }
```

Noções básicas sobre o rastreamento

Veja a seguir um segmento de rastreamento para um estágio do API Gateway. Para obter uma explicação detalhada sobre os campos que compõem o segmento de rastreamento, consulte [Documentos de segmento do AWS X-Ray](#) no Guia do desenvolvedor do AWS X-Ray.

```
{
    "Document": {
        "id": "a1b2c3d4a1b2c3d4",
        "name": "testxray/dev",
        "start_time": 1533928226.229,
        "end_time": 1533928226.614,
        "metadata": {
            "default": {
                "extended_request_id": "abcde12345abcde=",
                "request_id": "a1b2c3d4-a1b2-a1b2-a1b2-a1b2c3d4e5f6"
            }
        },
        "http": {
            "request": {
                "url": "https://example.com/dev?username=demo&message=hellofromdemo/",
                "method": "GET",
                "client_ip": "192.0.2.0",
                "x_forwarded_for": true
            },
            "response": {
                "status": 200,
                "content_length": 0
            }
        },
        "aws": {
            "xray": {
                "sampling_rule_name": "Default"
            },
            "api_gateway": {
                "account_id": "123412341234",
                "rest_api_id": "a1b2c3d4e5",
                "stage": "dev",
                "request_id": "a1b2c3d4-a1b2-a1b2-a1b2-a1b2c3d4e5f6"
            }
        },
        "annotations": {
            "aws:api_id": "a1b2c3d4e5",
            "aws:api_stage": "dev"
        },
        "trace_id": "1-a1b2c3d4-a1b2c3d4a1b2c3d4a1b2c3d4",
        "origin": "AWS::ApiGateway::Stage",
        "resource_arn": "arn:aws:apigateway:us-east-1::/restapis/a1b2c3d4e5/stages/dev",
        "subsegments": [
            {
                "id": "abcdefg12345678",
                "id": "abcdefg12345678"
            }
        ]
    }
}
```

```
"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"
}
```

Registrar chamadas para APIs do Amazon API Gateway com AWS CloudTrail

O Amazon API Gateway é integrado ao AWS CloudTrail, um serviço que fornece um registro de ações realizadas por um usuário, uma função ou um serviço da AWS no API Gateway. O CloudTrail captura todas as chamadas de API REST do API Gateway como eventos, incluindo as chamadas do console do API Gateway e as chamadas de código para as APIs do API Gateway. Se você criar uma trilha, você poderá habilitar a entrega contínua de eventos do CloudTrail para um bucket do Amazon S3, incluindo eventos para o API Gateway. Se não configurar uma trilha, você ainda poderá visualizar os eventos mais recentes no console do CloudTrail em Event history. Com as informações coletadas pelo CloudTrail, você pode 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 saber mais sobre CloudTrail, consulte o [AWS CloudTrail User Guide](#).

Informações sobre o API Gateway no CloudTrail

O CloudTrail está habilitado na sua conta da AWS ao criá-la. Quando ocorre uma atividade no Amazon API Gateway, ela é registrada em um evento do CloudTrail junto com outros eventos de serviços da AWS em Event history (Histórico de eventos). Você pode visualizar, pesquisar e fazer download de eventos recentes em sua conta da AWS. Para obter mais informações, consulte [Visualizar eventos com o histórico de eventos do CloudTrail](#).

Para obter um registro contínuo de eventos em sua conta da AWS, incluindo eventos do API Gateway, crie uma trilha. Uma trilha permite CloudTrail para fornecer arquivos de log a um bucket do Amazon S3. Por padrão, quando você cria uma trilha no console, ela é aplicada a todas as regiões. A trilha registra eventos em log de todas as regiões na partição da AWS e entrega os arquivos de log ao bucket do Amazon S3 que você especificar. Além disso, é possível configurar outros serviços da AWS para analisar mais profundamente e agir sobre os dados de evento coletados nos logs do CloudTrail. Para obter mais informações, consulte:

- Visão geral da criação de uma trilha
- CloudTrail Serviços compatíveis e integrações do
- Configuração de notificações do Amazon SNS para o CloudTrail
- Recebimento de arquivos de log do CloudTrail de várias regiões e Recebimento de arquivos de log do CloudTrail de várias contas

Todas as ações do Amazon API Gateway são registradas em log pelo CloudTrail e documentadas no [Referências de API na V1 e V2 do API Gateway \(p. 724\)](#). Por exemplo, as chamadas para criar uma nova API, recurso ou método no API Gateway geram entradas nos arquivos de log do CloudTrail.

Cada entrada de log ou evento contém informações sobre quem gerou a solicitação. As informações de identidade ajudam a determinar:

- Se a solicitação foi feita com credenciais de usuário da raiz ou do IAM.
- Se a solicitação foi feita com credenciais de segurança temporárias de uma função ou de um usuário federado.
- Se a solicitação foi feita por outro serviço da AWS.

Para obter mais informações, consulte [Elemento userIdentity do CloudTrail](#).

Noções básicas das entradas dos arquivos de log do API Gateway

Uma trilha é uma configuração que permite a entrega de eventos como arquivos de log em um bucket do Amazon S3 que você especificar. Os arquivos de log do CloudTrail contêm uma ou mais entradas de log. Um evento representa uma única solicitação de qualquer origem e inclui informações sobre a ação solicitada, a data e hora da ação, parâmetros de solicitação e assim por diante. Os arquivos de log do CloudTrail não são um rastreamento de pilha ordenada das chamadas da API pública. Assim, elas não são exibidas em nenhuma ordem específica.

O exemplo a seguir mostra uma entrada de log do CloudTrail que demonstra a ação de obtenção de recurso 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 ...  
]  
}
```

Monitorar a execução da API com o Amazon CloudWatch

Você pode 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 automaticamente enviados ao 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 você pode analisar de diferentes maneiras. A lista abaixo mostra alguns usos comuns das métricas. Essas são sugestões para você começar, e não uma lista abrangente.

- Monitore as métricas de IntegrationLatency para medir a capacidade de resposta do back-end.
- Monitore as métricas de Latency para medir a capacidade de resposta geral das suas chamadas de API.
- Monitor as métricas de CacheHitCount e CacheMissCount para otimizar capacidades de cache de modo a alcançar o desempenho desejado.

Tópicos

- [Amazon API Gateway Dimensões e métricas \(p. 599\)](#)
- [Visualizar métricas do CloudWatch com o painel da API no API Gateway \(p. 602\)](#)
- [Visualizar métricas do API Gateway no console do CloudWatch \(p. 602\)](#)
- [Visualizar eventos de log do API Gateway no console do CloudWatch \(p. 603\)](#)
- [Ferramentas de monitoramento na AWS \(p. 603\)](#)

Amazon API Gateway Dimensões e métricas

As métricas e dimensões que o API Gateway envia ao Amazon CloudWatch estão listadas abaixo. Para obter mais informações, consulte [Monitorar a execução da API com o Amazon CloudWatch \(p. 599\)](#).

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>A estatística <code>Sum</code> representa essa métrica, ou seja, a contagem total de erros <code>4XXError</code> no período especificado. A estatística <code>Average</code> representa a taxa de erros <code>4XXError</code>, ou seja, a contagem total de erros <code>4XXError</code> dividida pelo número total de solicitações durante o período. O denominador corresponde à métrica <code>Count</code> (abaixo).</p> <p>Unit: Count</p>
5XXError	<p>O número de erros no lado do servidor capturados em um determinado período.</p> <p>A estatística <code>Sum</code> representa essa métrica, ou seja, a contagem total de erros <code>5XXError</code> no período especificado. A estatística <code>Average</code> representa a taxa de erros <code>5XXError</code>, ou seja, a contagem total de erros <code>5XXError</code> dividida pelo número total de solicitações durante o período. O denominador corresponde à métrica <code>Count</code> (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 <code>Sum</code> representa essa métrica, ou seja, a contagem total de acertos de cache no período especificado. A estatística <code>Average</code> 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 <code>Count</code> (abaixo).</p> <p>Unit: Count</p>
CacheMissCount	<p>O número de solicitações atendidas pelo back-end em um determinado período quando o armazenamento em cache da API está habilitado.</p> <p>A estatística <code>Sum</code> representa essa métrica, ou seja, a contagem total de erros de cache no período especificado. A estatística <code>Average</code> 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 <code>Count</code> (abaixo).</p> <p>Unit: Count</p>
Count	<p>O número total de solicitações de API em um determinado período.</p> <p>A estatística <code>SampleCount</code> representa essa métrica.</p>

Métrica	Descrição
	Unit: Count
IntegrationLatency	O tempo entre o API Gateway retransmitir uma solicitação para o back-end e receber uma resposta do back-end. Unit: Millisecond
Latency	O tempo desde quando o API Gateway recebe uma solicitação de um cliente e retorna uma resposta para o cliente. A latência inclui a latência de integração e outras sobrecargas do API Gateway. Unit: Millisecond

Dimensões para as métricas

Você pode 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	Filtrar as métricas do API Gateway para a API REST com o nome da API especificado.
ApiName, Method, Resource, Stage	Filtrar as métricas do API Gateway para o método de API com o nome da API, estágio, recurso e método especificados. O API Gateway não enviará essas métricas, a menos que você habilite explicitamente as métricas detalhadas do CloudWatch. Você pode fazer isso no console selecionando Enable CloudWatch Metrics (Habilitar métricas do CloudWatch) em um estágio na guia Settings (Configurações). Como alternativa, você pode chamar o comando update-stage da AWS CLI para atualizar a propriedade metricsEnabled para true. Habilitar essas métricas incorrerá em cobranças adicionais na conta. Para obter informações sobre a definição de preço, consulte Definição de preço do Amazon CloudWatch .
ApiName, Stage	Filtrar as métricas do API Gateway para o recurso de estágio de API com o nome e o estágio de API especificados.

Visualizar métricas do CloudWatch com o painel da API no API Gateway

Você pode usar o painel da API no console do API Gateway para exibir as métricas do CloudWatch da sua API implantada no API Gateway. Elas são apresentadas como um resumo da atividade da API ao longo do tempo.

Tópicos

- [Pré-requisitos \(p. 602\)](#)
- [Examinar atividades da API no painel \(p. 602\)](#)

Pré-requisitos

1. Você deve ter uma API criada no API Gateway. Siga as instruções em [Criação de uma API REST no Amazon API Gateway \(p. 182\)](#).
2. Você deve ter a API implantada pelo menos uma vez. Siga as instruções em [Implantando uma API REST no Amazon API Gateway \(p. 516\)](#).
3. Para obter métricas do CloudWatch para métodos individuais, você deve ter Logs do CloudWatch habilitados para esses métodos em determinado estágio, tal como descrito em [Atualizar configurações de estágio \(p. 521\)](#). Sua conta será cobrada pelo acesso aos logs em nível de métodos, mas não pelo acesso aos logs em nível de API ou estágio.

Examinar atividades da API no painel

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha o nome da API.
3. Na API selecionada, escolha Dashboard (Painel).
4. Para exibir um resumo das atividades da API ao longo do tempo, para Stage (Estágio), escolha o estágio desejado.
5. Use From (De) e To (Para) para inserir o intervalo de datas.
6. Atualize, se necessário, e visualize as métricas individuais exibidas em gráficos separados intitulados API Calls (Chamadas da API), Integration Latency (Latência de integração), Latency (Latência), 4xx Error (Erro 4xx) e 5xx Error (Erro 5xx). Os gráficos CacheHitCount e CacheMissCount serão exibidos somente se o armazenamento em cache de APIs tiver sido habilitado.

Tip

Para examinar métricas do CloudWatch em nível de método, certifique-se de ter habilitado Logs do CloudWatch em nível de método. Para obter mais informações sobre como configurar logs em nível de método, consulte [Atualizar configurações de estágio usando o console do API Gateway \(p. 521\)](#).

Visualizar 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 do API Gateway no console do CloudWatch

1. Abra o console do CloudWatch em <https://console.aws.amazon.com/cloudwatch/>.
2. Se necessário, altere a região. 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 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, em seguida, selecione APIs e nomes de métrica desejados.
6. Para visualizar as métricas por API específica, escolha o painel By Api Name (Por nome de API). E, em seguida, selecione APIs e nomes de métrica desejados.

Para visualizar métricas usando o AWS CLI

1. Em um prompt de comando, use o seguinte comando para listar métricas:

```
aws cloudwatch list-metrics --namespace "AWS/ApiGateway"
```

2. Para exibir as estatísticas específicas (por exemplo, Average) por um período de intervalos de 5 minutos, chame o seguinte comando:

```
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
```

Visualizar eventos de log do API Gateway no console do CloudWatch

Para visualizar solicitações e respostas de APIs registradas usando o console do CloudWatch

1. No painel de navegação, selecione Logs.
2. Na tabela Log Groups (Grupos de logs), selecione um grupo de logs do nome API-Gateway-Execution-Logs_{rest-api-id}/{stage-name}.
3. 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.
4. 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 grupos ou fluxos de log. Não exclua manualmente os fluxos ou grupos de log da API do API Gateway. Deixe que o API Gateway gerencie 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

A AWS fornece várias ferramentas que você pode usar 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

Você pode usar as seguintes ferramentas de monitoramento automatizado para observar o API Gateway e relatar quando algo estiver errado:

- Amazon CloudWatch Alarms (Alarmes do CW) – 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 determinado limite ao longo de vários períodos de tempo. A ação é uma notificação enviada a um tópico do Amazon Simple Notification Service (Amazon SNS) ou a uma política do Amazon EC2 Auto Scaling. Os alarmes do CloudWatch não invocam ações simplesmente porque estão 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 com o Amazon CloudWatch \(p. 599\)](#).
- Amazon CloudWatch Logs – monitore, armazene e acesse seus arquivos de log de instâncias do AWS CloudTrail ou de outras origens. Para mais informações, consulte [Monitoramento de arquivos de log](#) no Guia do usuário do Amazon CloudWatch.
- Eventos do Amazon CloudWatch – Faça correspondência de eventos e direcione-os a uma ou mais funções ou fluxos de destino para fazer alterações, capturar informações de estado e realizar ações corretivas. Para obter mais informações, consulte [O que é o Amazon CloudWatch Events](#) no Guia do usuário do Amazon CloudWatch.
- Monitoramento do log do 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 aplicativos de processamento de logs em Java e confirme se os arquivos de log não foram alterados após a distribuição pelo CloudTrail. Para obter mais informações, consulte [Trabalho com arquivos de log do CloudTrail](#) no AWS CloudTrail User Guide.

Ferramentas de monitoramento manual

Outra parte importante do monitoramento do API Gateway envolve o monitoramento manual desses itens que os alarmes do CloudWatch não abrangem. O API Gateway, o CloudWatch e outros painéis do console da AWS fornecem uma visão rápida do estado do ambiente da AWS. Recomendamos que você também verifique os arquivos de log na execução da API.

- O painel API Gateway mostra as seguintes estatísticas para determinado 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 o seguinte:
 - Alertas e status atual
 - Gráficos de alertas e recursos
 - Estado de integridade do serviço

Além disso, você pode usar o CloudWatch para fazer o seguinte:

- Criar [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 de Amazon SNS quando o alarme mudar 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 por uma política do Auto Scaling. Os alarmes invocam ações apenas para alterações de estado sustentado. Os alarmes do CloudWatch não invocam ações simplesmente porque estão em um estado específico. O estado deve ter sido alterado e mantido por um número específico de períodos.

Usar o Amazon Kinesis Data Firehose como destino para registro de acesso em logs do API Gateway

Para ajudar a depurar problemas relacionados ao acesso do cliente à sua API, é possível registrar chamadas de API para o Amazon Kinesis Data Firehose. Para obter mais informações sobre como usar o Kinesis Data Firehose, consulte [O que é o Amazon Kinesis Data Firehose?](#).

Para o registro de acesso em logs, só é possível habilitar o CloudWatch ou o Kinesis Data Firehose — não é possível habilitar ambos. No entanto, é possível habilitar o CloudWatch para registro de execução em logs e o Kinesis Data Firehose para registro de acesso em logs.

Tópicos

- [Formatos de logs do Kinesis Data Firehose para o API Gateway \(p. 605\)](#)
- [Permissões de log do Kinesis Data Firehose \(p. 605\)](#)
- [Configurar o registro de acesso em logs do Kinesis Data Firehose usando o console do API Gateway \(p. 605\)](#)

Formatos de logs do Kinesis Data Firehose para o API Gateway

O registro em log do Kinesis Data Firehose usa o mesmo formato que o [registro em log do CloudWatch](#).

Permissões de log do Kinesis Data Firehose

Quando o registro de acesso em logs do Kinesis Data Firehose estiver habilitado em um estágio, o API Gateway criará uma função vinculada a um serviço na sua conta se a função ainda não existir. 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 de entrega do Kinesis Data Firehose deve ser `amazon-apigateway-{your-delivery-stream-name}`.

Configurar o registro de acesso em logs do Kinesis Data 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 de entrega do Kinesis Data Firehose.

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Faça uma das coisas a seguir:
 - a. Escolha uma API existente e, em seguida, escolha um estágio.
 - b. Crie uma API e implante-a em um estágio.

3. Escolha Logs/Tracing no Stage Editor.
4. Para habilitar o registro de acesso em logs a um fluxo de entrega do Kinesis Data Firehose:
 - a. Selecione Enable Access Logging (Habilitar o registro de acesso em logs) em Custom Access Logging (Registro de acesso personalizado em logs).
 - b. Insira o ARN de um fluxo de entrega do Kinesis Data Firehose em Access Log Destination ARN (ARN do destino do log de acesso). O formato do ARN é:
`arn:aws:firehose:{region}:{account-id}:deliverystream:amazon-apigateway-{your-delivery-stream-name}`.

Note

O nome do fluxo de entrega deve ser `amazon-apigateway-{your-delivery-stream-name}`.

- c. Insira um formato de log em Log Format (Formato do log). Você pode escolher CLF, JSON, XML ou CSV para usar um dos exemplos fornecidos como um guia.
5. Escolha Save Changes (Salvar alterações).

O API Gateway já está pronto para registrar em log as solicitações à sua API no Kinesis Data Firehose. Não é necessário reimplantar a API ao atualizar as configurações do estágio, os logs ou as variáveis do estágio.

Extensões do API Gateway para OpenAPI

As extensões do API Gateway oferecem suporte à autorização específica da AWS e às integrações de API específicas do API Gateway. Nesta seção, descreveremos as extensões do API Gateway para a especificação OpenAPI de APIs REST.

Tip

Para compreender como as extensões do API Gateway são usadas em um aplicativo, você pode usar o console do API Gateway para criar uma API e exportá-la para um arquivo de definição do OpenAPI. Para obter mais informações sobre como exportar uma API, consulte [Exportar uma API REST \(p. 700\)](#).

Tópicos

- [Objeto x-amazon-apigateway-any-method \(p. 607\)](#)
- [Propriedade x-amazon-apigateway-api-key-source \(p. 608\)](#)
- [Objeto x-amazon-apigateway-auth \(p. 608\)](#)
- [Objeto x-amazon-apigateway-authorizer \(p. 609\)](#)
- [Propriedade x-amazon-apigateway-authtype \(p. 612\)](#)
- [Propriedade x-amazon-apigateway-binary-media-types \(p. 613\)](#)
- [Objeto x-amazon-apigateway-documentation \(p. 613\)](#)
- [Objeto x-amazon-apigateway-endpoint-configuration \(p. 614\)](#)
- [Objeto x-amazon-apigateway-gateway-responses \(p. 615\)](#)
- [Objeto x-amazon-apigateway-gateway-responses.gatewayResponse \(p. 615\)](#)
- [Objeto x-amazon-apigateway-gateway-responses.responseParameters \(p. 616\)](#)
- [Objeto x-amazon-apigateway-gateway-responses.responseTemplates \(p. 617\)](#)
- [Objeto x-amazon-apigateway-integration \(p. 617\)](#)
- [Objeto x-amazon-apigateway-integration.requestTemplates \(p. 621\)](#)
- [Objeto x-amazon-apigateway-integration.requestParameters \(p. 622\)](#)

- Objeto `x-amazon-apigateway-integration.responses` (p. 623)
- Objeto `x-amazon-apigateway-integration.response` (p. 624)
- Objeto `x-amazon-apigateway-integration.responseTemplates` (p. 625)
- Objeto `x-amazon-apigateway-integration.responseParameters` (p. 626)
- Propriedade `x-amazon-apigateway-request-validator` (p. 626)
- Objeto `x-amazon-apigateway-request-validators` (p. 627)
- Objeto `x-amazon-apigateway-request-validation.validators.requestValidator` (p. 628)

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.

Properties

Nome da propriedade	Type	Descrição
<code>x-amazon-apigateway-integration</code>	Objeto x-amazon-apigateway-integration (p. 617)	Especifica a integração do método com o back-end. Esta é uma propriedade estendida do objeto de Operação do OpenAPI . A integração pode ser de tipo AWS, AWS_PROXY, HTTP, HTTP_PROXY ou MOCK.

Exemplo 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",
      "passthroughBehavior": "when_no_match",
      "httpMethod": "POST",
      "type": "aws_proxy"
    }
  }
}
```

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`.

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",
  .
  .
  .
}
```

Objeto x-amazon-apigateway-auth

Define um tipo de autenticação a ser aplicado para a autenticação de invocações de método no API Gateway.

Properties

Nome da propriedade	Type	Descrição
<code>type</code>	<code>string</code>	Especifica o tipo de autenticação. Especifique <code>"NONE"</code> para acesso aberto. Especifique <code>"AWS_IAM"</code> para usar permissões do IAM. Os valores não diferenciam letras maiúsculas de minúsculas.

Exemplo de x-amazon-apigateway-auth

O exemplo a seguir define o tipo de autenticação como um método de API REST.

OpenAPI 3.0.1

```
{
  "openapi": "3.0.1",
  "info": {
```

```

        "title": "openapi3",
        "version": "2018-12-04T05:22:50Z"
    },
    "servers": [
        {
            "url": "https://hs8l62bm7l.execute-api.us-west-2.amazonaws.com/{basePath}",
            "variables": {
                "basePath": {
                    "default": "/dev"
                }
            }
        }
    ],
    "security": [
        {
            "api_key": []
        }
    ],
    "paths": {
        "/key": {
            "get": {
                "x-amazon-apigateway-auth": {
                    "type": "NONE"
                },
                "security": [
                    {
                        "api_key": []
                    }
                ]
            }
        },
        "components": {
            "securitySchemes": {
                "api_key": {
                    "type": "apiKey",
                    "name": "x-api-key",
                    "in": "header"
                }
            }
        }
    }
}
    
```

Objeto x-amazon-apigateway-authorizer

Define um autorizador do Lambda (anteriormente conhecido como autorizador personalizado) a ser aplicado para autorização de invocações de método no API Gateway. Este objeto é uma propriedade estendida do objeto de [Definições de segurança do OpenAPI](#).

Properties

Nome da propriedade	Type	Descrição
type	string	O tipo de autorizador. Essa é uma propriedade obrigatória. Especifique "token" para um autorizador com a identidade do chamador incorporada em um token de autorização. Especifique "request" para um autorizador com a identidade

Nome da propriedade	Type	Descrição
		do chamador contida nos parâmetros da solicitação.
authorizerUri	string	O URI (Uniform Resource Identifier) da função Lambda do autorizador. A sintaxe é a seguinte: <div style="border: 1px solid black; padding: 5px;"><code>"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"</code></div>
authorizerCredentials	string	As credenciais necessárias para invocar o autorizador, se houver, no formato de um ARN de uma função de execução do IAM. Por exemplo, "arn:aws:iam:: account-id : IAM_role ".
identitySource	string	Lista de expressões de mapeamento separadas por vírgulas dos parâmetros de solicitação como a origem de identidade. Aplicável para o autorizador somente do tipo "solicitação".
identityValidationExpression	string	Uma expressão regular para validar o token como identidade de entrada. Por exemplo, "^x-[az]+".
authorizerResultTtlInSeconds	string	O número de segundos durante os quais a política IAM resultante é armazenada em cache.

Exemplos de x-amazon-apigateway-authorizer

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",                                // Required and the value must be "apiKey"
        for an API Gateway API.
        "name" : "Authorization",                        // The name of the header containing the
        authorization token.
        "in" : "header",                                 // Required and the value must be "header"
        for an API Gateway API.
        "x-amazon-apigateway-authtype" : "oauth2", // Specifies the authorization mechanism
        for the client.
    }
}
  
```

```

    "x-amazon-apigateway-authorizer" : {           // An API Gateway Lambda authorizer
        "definition"
            "type" : "token",                      // Required property and the value must
        "token"
            "authorizerUri" : "arn:aws:apigateway:us-east-1:lambda:path/2015-03-31/functions/
arn:aws:lambda:us-east-1:account-id:function:function-name/invocations",
            "authorizerCredentials" : "arn:aws:iam::account-id:role",
            "identityValidationExpression" : "^x-[a-z]+",
            "authorizerResultTtlInSeconds" : 60
        }
    }
}

```

O seguinte trecho de objeto de operação do OpenAPI define o GET /http para usar o autorizador do Lambda especificado acima.

```

"/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
        }
    }
}

```

O exemplo de definições de segurança do OpenAPI a seguir especifica 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
        }
    }
}
```

O exemplo de definições de segurança do OpenAPI a seguir especifica um autorizador do Lambda do API Gateway, 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
        }
    }
}
```

Propriedade x-amazon-apigateway-authtype

Especifique as informações opcionais definidas pelo cliente que descrevem um autorizador do Lambda (anteriormente conhecido como autorizador personalizado). Elas são usadas para que a API do API Gateway importe e exporte sem impacto funcional.

Esta propriedade é uma propriedade estendida do objeto de [Operação de definições de segurança do OpenAPI](#).

Exemplo de x-amazon-apigateway-authtype

O exemplo a seguir define o tipo de um autorizador do Lambda usando OAuth 2.

```
"cust-authorizer" : {
    "type" : "...", // required
    "name" : "...", // name of the identity source header
    "in" : "header", // must be header
    "x-amazon-apigateway-authtype" : "oauth2", // Specifies the authorization mechanism
for the client.
    "x-amazon-apigateway-authorizer" : {
        ...
    }
}
```

O seguinte exemplo de definição de segurança especifica uma autorização usando protocolo [Signature Version 4 da AWS](#):

```
"sigv4" : {
    "type" : "apiKey",
    "name" : "Authorization",
    "in" : "header",
    "x-amazon-apigateway-authtype" : "awsSigv4"
}
```

Consulte também

[authorizer.authType](#)

Propriedade x-amazon-apigateway-binary-media-types

Especifica a lista de tipos de mídia binários que receberão suporte do API Gateway, como application/octet-stream, image/jpeg etc. Esta 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.

Properties

Nome da propriedade	Type	Descrição
documentationParts	Array	Uma matriz das instâncias de DocumentationPart exportadas ou importadas.

Nome da propriedade	Type	Descrição
version	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 de 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
      }
    ]
  }
}
```

Objeto x-amazon-apigateway-endpoint-configuration

Especifica detalhes da configuração do endpoint usada por essa API Rest. Esta extensão é uma propriedade estendida do objeto de [Operação do OpenAPI](#). Esse objeto deve estar presente em [extensões de fornecedores de nível superior](#) para Swagger2.0, enquanto para OpenAPI3.0, ele deve estar presente nas extensões de fornecedores do [objeto Server](#).

Properties

Nome da propriedade	Type	Descrição
vpcEndpointIds	Uma matriz de String	Uma lista de identificadores VpcEndpoint nos quais criar aliases do Route 53 para uma API REST. Só é compatível com o tipo de endpoint (<code>PRIVATE</code>).

Exemplo 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" ]
```

}

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.

Properties

Nome da propriedade	Type	Descrição
responseType	x-amazon-apigateway-gateway-responses.gatewayResponse (p. 615)	Um GatewayResponse para o responseType especificado.

Exemplo de x-amazon-apigateway-gateway-responses

A seguinte extensão do API Gateway para o OpenAPI define um mapa [GatewayResponses](#) contendo 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.

Properties

Nome da propriedade	Type	Descrição
responseParameters	x-amazon-apigateway-gateway-responses.responseParameters (p. 610)	Especifica os parâmetros GatewayResponse , ou seja, os parâmetros de cabeçalho. Os valores de parâmetros podem usar qualquer valor de parâmetro de solicitação (p. 301) de entrada ou um valor personalizado estático.

Nome da propriedade	Type	Descrição
<code>responseTemplates</code>	<code>x-amazon-apigateway-gateway- responses.responseTemplates</code> (p. 617)	Especifica os modelos de mapeamento da resposta de gateway. Os modelos não são processados pelo mecanismo VTL.
<code>statusCode</code>	<code>string</code>	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 o OpenAPI define um `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.

Properties

Nome da propriedade	Type	Descrição
<code>gatewayresponse.param-position.param-name</code>	<code>string</code>	<code>param-position</code> pode ser <code>header</code> , <code>path</code> ou <code>querystring</code> . Para obter mais informações, consulte Mapear dados de solicitação de método para parâmetros de solicitação de integração (p. 301).

Exemplo de x-amazon-apigateway-gateway- responses.responseParameters

O seguinte exemplo de extensões do OpenAPI mostra uma expressão de mapeamento de parâmetros de resposta `GatewayResponse` para habilitar o suporte 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 de chave/valor, a chave é o tipo de conteúdo; por exemplo, "application/json", e o valor é um modelo de mapeamento transformado em string para substituições de variáveis simples. Um modelo de mapeamento [GatewayResponse](#) não é processado pelo mecanismo [Velocity Template Language \(VTL\)](#).

Properties

Nome da propriedade	Type	Descrição
<code>content-type</code>	string	Um modelo de mapeamento de corpo GatewayResponse que oferece suporte apenas à substituição de variáveis simples para personalizar um corpo de resposta de gateway.

Exemplo de x-amazon-apigateway-gateway- responses.responseTemplates

O seguinte exemplo de extensões do OpenAPI mostra um modelo de mapeamento [GatewayResponse](#) para personalizar uma resposta de erro gerada pelo API Gateway em um formato específico de aplicativo.

```
"responseTemplates": {  
    "application/json": "{ \"message\": \"$context.error.messageString\", \"type\": \"$context.error.responseType\", \"statusCode\": '488' }"  
}
```

O seguinte exemplo de extensões do OpenAPI 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' }"  
}
```

Objeto x-amazon-apigateway-integration

Especifica detalhes da integração de back-end 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](#).

Properties

Nome da propriedade	Type	Descrição
<code>cacheKeyParameters</code>	Uma matriz de <code>string</code>	Uma lista de parâmetros de solicitação cujos valores devem ser armazenados em cache.
<code>cacheNamespace</code>	<code>string</code>	Um grupo de tags específicas de API dos parâmetros em cache relacionados.
<code>connectionId</code>	<code>string</code>	O ID de um VpcLink para a integração privada.
<code>connectionType</code>	<code>string</code>	O tipo de conexão da integração. O valor válido é " <code>VPC_LINK</code> " para integração privada ou " <code>INTERNET</code> ", para outras.
<code>credentials</code>	<code>string</code>	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 manualmente para permitir que a API accesse o recurso. Para obter mais informações, consulte Concessão de permissões com o uso de uma política de recursos . Observação: para ter o melhor desempenho possível, ao usar credenciais do IAM, certifique-se de que endpoints regionais STS da AWS estejam habilitados para a região na qual essa API está implantada.
<code>contentHandling</code>	<code>string</code>	Tipos de conversão de codificação da carga de solicitação. Os valores válidos são 1) <code>convert_to_text</code> , para converter uma carga binária em uma string codificada em Base64, converter uma carga de texto em uma string codificada em <code>utf-8</code> ou transferir a carga de texto nativamente sem modificação, e 2) <code>convert_to_binary</code> , para converter uma carga de texto em um blob decodificado em Base64 ou transferir uma carga binária nativamente sem modificação.

Nome da propriedade	Type	Descrição
<code>httpMethod</code>	<code>string</code>	O método HTTP usado na solicitação de integração. Para invocações de função do Lambda, o valor deve ser <code>POST</code> .
<code>passthroughBehavior</code>	<code>string</code>	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 Integration.passthroughBehavior .
<code>requestParameters</code>	Objeto x-amazon-apigateway-integration.requestParameters (p. 622)	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> .
<code>requestTemplates</code>	Objeto x-amazon-apigateway-integration.requestTemplates (p. 620)	Modelos de mapeamento para uma carga de solicitação de tipos MIME especificados.
<code>responses</code>	Objeto x-amazon-apigateway-integration.responses (p. 623)	Define as respostas do método e especifica mapeamentos de parâmetros desejados ou mapeamentos de carga de respostas de integração para respostas de método.
<code>timeoutInMillis</code>	<code>integer</code>	Tempos limite para a integração entre 50 ms e 29.000 ms.

Nome da propriedade	Type	Descrição
<code>type</code>	<code>string</code>	<p>O tipo de integração com o back-end especificado. O valor válido é</p> <ul style="list-style-type: none"> • <code>http</code> ou <code>http_proxy</code>: para integração com um back-end HTTP;; • <code>aws_proxy</code>: para integração com funções do Lambda da AWS; • <code>aws</code>: para integração com funções do Lambda da AWS ou outros serviços da AWS, como Amazon DynamoDB, Amazon Simple Notification Service ou Amazon Simple Queue Service; • <code>mock</code>: para integração com o API Gateway sem invocar qualquer back-end. <p>Para obter mais informações sobre os tipos de integração, consulte integration:type.</p>
<code>uri</code>	<code>string</code>	O URI de endpoint do back-end. Para integrações do tipo <code>aws</code> , este é um valor de ARN. Para a integração HTTP, esta é a URL do endpoint HTTP, incluindo o esquema <code>https</code> ou <code>http</code> .

Exemplo de x-amazon-apigateway-integration

O exemplo a seguir integra o método POST de uma API com uma função do Lambda no back-end. 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> "
    },
}
```

```

"requestParameters" : {
    "integration.request.path.stage" : "method.request.querystring.version",
    "integration.request.querystring.provider" : "method.request.querystring.vendor"
},
"cacheNamespace" : "cache namespace",
"cacheKeyParameters" : [],
"responses" : {
    "2\\d{2}" : {
        "statusCode" : "200",
        "responseParameters" : {
            "method.response.header.requestId" : "integration.response.header.cid"
        },
        "responseTemplates" : {
            "application/json" : "#set ($root=$input.path('$')) { \"stage\": \"$root.name\", \"$user-id\": \"$root.key\" }",
            "application/xml" : "#set ($root=$input.path('$')) <stage>$root.name</
stage> "
        }
    },
    "302" : {
        "statusCode" : "302",
        "responseParameters" : {
            "method.response.header.Location" :
                "integration.response.body.redirect.url"
        }
    },
    "default" : {
        "statusCode" : "400",
        "responseParameters" : {
            "method.response.header.test-method-response-header" : "'static value'"
        }
    }
}
}

```

Observe que as aspas duplas ("") da string JSON nos modelos de mapeamento devem ser escapadas em string (\").

Objeto x-amazon-apigateway-integration.requestTemplates

Especifica modelos de mapeamento para uma carga de solicitação dos tipos MIME especificados.

Properties

Nome da propriedade	Type	Descrição
MIME type	string	Um exemplo do tipo MIME é application/json. Para obter informações sobre como criar um modelo de mapeamento, consulte Modelos de mapeamento (p. 277).

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

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.

Properties

Nome da propriedade	Type	Descrição
<code>integration.request.<param-type>.<param-name></code>	string	Normalmente, o valor é um parâmetro de solicitação de método predefinido do formato <code>method.request.<param-type>.<param-name></code> , em que <code><param-type></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><param-name></code> é uma expressão de caminho JSON sem o prefixo <code>\$</code> .

x-amazon-apigateway-integration.requestParameters Exemplo

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 via OpenAPI ou AWS 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.

Properties

Nome da propriedade	Type	Descrição
<i>Padrão do status de resposta</i>	Objeto x-amazon-apigateway-integration.response (p. 624)	<p>Expressão regular de seleção usada para corresponder a resposta de integração com a resposta de método. Para integrações HTTP, essa expressão regular aplica-se o código de status da resposta de integração. Para invocações do Lambda, a expressão regular aplica-se ao campo <code>errorMessage</code> do objeto de informações de erro retornado pelo AWS Lambda como um corpo de resposta de falha quando a exceção da função do Lambda lança uma exceção.</p> <p>Note</p> <p>O nome da propriedade do <i>Padrão do status de resposta</i> refere-se a um código de status de resposta ou a uma expressão regular que descreve um grupo de códigos de status de resposta. Ele não corresponde a nenhum identificador de um recurso IntegrationResponse na API REST do API Gateway.</p>

x-amazon-apigateway-integration.responses Exemplo

O exemplo a seguir mostra uma lista de respostas 2xx e 302. Para a resposta 2xx, a resposta do método é mapeada a partir da carga da resposta de integração do tipo MIME `application/json` ou `application/xml`. Essa resposta usa os modelos de mapeamento fornecidos. Para a resposta 302, a resposta de método retorna um cabeçalho `Location` cujo valor é derivado da propriedade `redirect.url` na carga da resposta de integração.

```
"responses" : {
```

```

    "2\\d{2}" : {
        "statusCode" : "200",
        "responseTemplates" : {
            "application/json" : "#set ($root=$input.path('$')) { \"stage\": \"$root.name\" , \"user-id\": \"$root.key\" }",
            "application/xml" : "#set ($root=$input.path('$')) <stage>$root.name</stage> "
        }
    },
    "302" : {
        "statusCode" : "302",
        "responseParameters" : {
            "method.response.header.Location" : "integration.response.body.redirect.url"
        }
    }
}

```

Objeto x-amazon-apigateway-integration.response

Define uma resposta e especifica mapeamentos de parâmetros ou mapeamentos de carga a partir da resposta de integração para a resposta de método.

Properties

Nome da propriedade	Type	Descrição
<code>statusCode</code>	<code>string</code>	Código de status HTTP para a resposta de método; por exemplo, "200". Isso deve equivaler a uma resposta correspondente no campo <code>responses</code> da Operação do OpenAPI .
<code>responseTemplates</code>	Objeto x-amazon-apigateway-integration.responseTemplates (p. 625)	Especifica modelos de mapeamento específicos de tipo MIME para a carga da resposta.
<code>responseParameters</code>	Objeto x-amazon-apigateway-integration.responseParameters (p. 626)	Especifica mapeamentos de parâmetros para a resposta. Somente os parâmetros <code>header</code> e <code>body</code> da resposta de integração podem ser mapeados para os parâmetros <code>header</code> do método.
<code>contentHandling</code>	<code>string</code>	Tipos de conversão de codificação da carga de resposta. Os valores válidos são 1) <code>CONVERT_TO_TEXT</code> , para converter uma carga binária em uma string codificada em Base64, converter uma carga de texto em uma string codificada em <code>utf-8</code> ou transferir a carga de texto nativamente sem modificação, e 2) <code>CONVERT_TO_BINARY</code> , para converter uma carga de texto em

Nome da propriedade	Type	Descrição
		um blob decodificado em Base64 ou transferir uma carga binária nativamente sem modificação.

x-amazon-apigateway-integration.response Exemplo

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 back-end. 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.

Properties

Nome da propriedade	Type	Descrição
MIME type	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 Modelos de mapeamento (p. 277) . Um exemplo do tipo MIME 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. Somente os tipos header e body dos parâmetros de resposta de integração podem ser mapeados para o tipo header da resposta de método.

Properties

Nome da propriedade	Type	Descrição
<code>method.response.header.<param name></code>	string	O valor do parâmetro nomeado pode ser derivado dos tipos header e body somente dos parâmetros de resposta de integração.

x-amazon-apigateway-integration.responseParameters Exemplo

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"
}
```

Propriedade x-amazon-apigateway-request-validator

Especifica um validador de solicitação, fazendo referência a um `request_validator_name` do mapa [Objeto x-amazon-apigateway-requestValidators \(p. 627\)](#), 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-validator Exemplo

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-requestValidators" : {
    "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-requestValidators

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 aplica-se a uma API.

Properties

Nome da propriedade	Type	Descrição
<code>request_validator_name</code>	Objeto x-amazon-apigateway-requestValidators.requestValidator (p. 628)	<p>Especifica as regras de validação que consistem no validador nomeado. Por exemplo:</p> <pre>"basic" : { "validateRequestBody" : true, "validateRequestParameters" : true },</pre> <p>Para aplicar esse validador a um método específico, faça referência ao nome do validador (<code>basic</code>) como o valor da propriedade Propriedade x-amazon-apigateway-requestvalidator (p. 626).</p>

x-amazon-apigateway-requestValidators Exemplo

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-validation" : {
    "basic" : {
        "validateRequestBody" : true,
        "validateRequestParameters" : true
    },
    "params-only" : {
        "validateRequestBody" : false,
        "validateRequestParameters" : true
    }
},
...
}
```

Objeto x-amazon-apigateway-request-validation.requestValidator

Especifica as regras de validação de um validador de solicitação como parte da definição do mapa [Objeto x-amazon-apigateway-request-validation \(p. 627\)](#).

Properties

Nome da propriedade	Type	Descrição
validateRequestBody	Boolean	Especifica se o corpo da solicitação deve ser validado (<code>true</code>) ou não (<code>false</code>).
validateRequestParameters	Boolean	Especifica se os parâmetros de solicitação necessários devem ser validados (<code>true</code>) ou não (<code>false</code>).

x-amazon-apigateway-request-validation.requestValidator Exemplo

O exemplo a seguir mostra um validador de solicitação somente para parâmetros:

```
"params-only": {
    "validateRequestBody" : false,
    "validateRequestParameters" : true
}
```

Criar, implantar e invocar uma API WebSocket no Amazon API Gateway

Para obter uma breve introdução ao suporte WebSocket no API Gateway, consulte [the section called “Use o API Gateway para criar APIs WebSocket” \(p. 3\)](#).

Tópicos

- [Sobre APIs WebSocket no API Gateway \(p. 629\)](#)
- [Criar uma API WebSocket no API Gateway \(p. 634\)](#)
- [Configurar rotas para uma API WebSocket no API Gateway \(p. 635\)](#)
- [Configurar integrações da API WebSocket no API Gateway \(p. 637\)](#)
- [Configurar respostas de rota para uma API WebSocket no API Gateway \(p. 642\)](#)
- [Implantar uma API WebSocket no API Gateway \(p. 643\)](#)
- [Invocar uma API WebSocket \(p. 645\)](#)
- [Controlar o acesso à API WebSocket no API Gateway \(p. 647\)](#)
- [Monitorar a execução da API WebSocket com CloudWatch \(p. 650\)](#)
- [Expressões de seleção do WebSocket no API Gateway \(p. 652\)](#)
- [Referência de modelos de mapeamento da API WebSocket do API Gateway \(p. 659\)](#)

Sobre APIs WebSocket no API Gateway

No API Gateway, você pode criar uma API WebSocket como um front-end stateful para um serviço da AWS (como Lambda ou DynamoDB) ou para um endpoint HTTP. A API WebSocket invoca o back-end 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 back-end. O back-end pode enviar mensagens de retorno para clientes conectados.

Na sua API WebSocket, mensagens JSON de entrada são direcionadas para integrações de back-end 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 “” \(p. 652\)](#).

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.

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 a integração é invocada.
- O API Gateway chama 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 “Gerenciamento de usuários conectados e aplicativos do cliente” \(p. 630\)](#).

Para obter mais informações sobre a rota `$default` e rotas personalizadas, consulte [the section called “Como invocar a integração de seu back-end” \(p. 631\)](#).

Os serviços de back-end podem enviar dados para aplicativos conectados do cliente. Para obter mais informações, consulte [the section called “Envio de dados dos serviços de back-end para clientes conectados” \(p. 634\)](#).

Gerenciamento de usuários conectados e aplicativos do cliente: Rotas `$connect` e `$disconnect`

Tópicos

- [A rota `\$connect` \(p. 630\)](#)
- [A rota `\$disconnect` \(p. 631\)](#)

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 `$connect` é opcional. No entanto, pode ser útil, visto que o back-end recebe o ID de usuário de clientes que se conectam. Você deve considerar configurar uma integração `$connect` se:

- Deseja ser notificado quando os clientes se conectarem e desconectarem.
- Deseja limitar as conexões ou controlar quem se conecta.
- Deseja que o back-end 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).

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 encerrada durante a execução, \$disconnect é considerado um evento de melhor esforço. O API Gateway envidará todos os esforços para fornecer o evento \$disconnect à sua integração, mas ele não pode garantir a entrega.

O back-end pode iniciar a desconexão ao utilizar a API @connections. Para obter mais informações, consulte [the section called “Use os comandos @connections em seu serviço de back-end” \(p. 646\)](#).

Como invocar a integração de seu back-end: Rotas e rotas personalizadas\$default

Tópicos

- [Usando rotas para processar mensagens \(p. 631\)](#)
- [A rota \\$default \(p. 632\)](#)
- [Rotas personalizadas \(p. 632\)](#)
- [Uso de integrações da API WebSocket do API Gateway para se conectar à sua lógica de negócios \(p. 633\)](#)
- [Diferenças importantes entre APIs WebSocket e APIs REST \(p. 633\)](#)
- [Tratamento de cargas binárias \(p. 634\)](#)

Usando rotas para processar mensagens

Nas APIs WebSocket do API Gateway, as mensagens podem ser enviadas do cliente para o serviço de back-end e vice-versa. Ao contrário do modelo de solicitação/resposta HTTP, no WebSocket é possível que o back-end 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 back-end pela rota \$default.

Note

O API Gateway oferece suporte a 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 “Tratamento de cargas binárias” \(p. 634\)](#).

Com APIs WebSocket no API Gateway, as mensagens JSON podem ser roteadas para executar um serviço de back-end 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. Você pode configurar a solicitação de rota para uma API WebSocket no console do API Gateway usando a AWS CLI ou usando 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.

Você pode 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 impede a solicitação sem chamar o back-end, envia uma resposta do "Bad request body" gateway semelhante à seguinte para o cliente e publica os resultados da validação no CloudWatch Logs:

```
{"message" : "Bad request body", "connectionId": "{connectionId}", "messageId": "{messageId}"}
```

Isso reduz as chamadas desnecessárias para o back-end 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 back-end sem receber uma resposta (comunicação unidirecional). No entanto, se você não fornecer uma resposta de rota, o API Gateway não enviará quaisquer informações sobre o resultado da sua integração aos clientes.

A rota \$default

Cada API WebSocket do API Gateway pode conter 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 back-end.
- 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 “Sobre APIs WebSocket” \(p. 629\)](#).)

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"}
```

Uso de 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, você deve especificar a integração que gostaria de utilizar. 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 back-end para processar a solicitação recebida de seu cliente. Uma resposta de integração contém os dados que o back-end retorna para o API Gateway e que podem ser usados para construir 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 “Configurar integrações da API WebSocket” \(p. 637\)](#).

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, você deve criar uma rota primeiro e, em seguida, 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 estreitamente se relacionam entre si, talvez você prefira 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-las uma vez e atribuí-las 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 \(p. 652\)](#) 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>, $integration.response.statusCode` e `$integration.response.header.<headerName>`.

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 back-end 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 back-end 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.

Tratamento de cargas binárias

As APIs WebSocket do API Gateway não oferecem suporte a 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 para converter `CONVERT_TO_BINARY` a carga de string codificada em Base64 para modo binário.

Para retornar uma resposta de rota para uma carga binária em integrações não proxy, você pode definir a propriedade `contentHandlingStrategy` da resposta de integração para converter `CONVERT_TO_TEXT` a carga do modo binário para string codificada em Base64.

Envio de dados dos serviços de back-end para clientes conectados

As APIs WebSocket do API Gateway oferecem as seguintes maneiras para enviar dados de serviços de back-end para clientes conectados:

- Uma integração pode enviar uma resposta, que é retornada 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 “Use os comandos `@connections` em seu serviço de back-end” \(p. 646\)](#).

Criar uma API WebSocket no API Gateway

Você pode criar uma API WebSocket no console do API Gateway ao utilizar o comando `create-api` da AWS CLI ou o comando `CreateApi` em um AWS SDK. Os procedimentos a seguir mostram como criar uma nova API WebSocket.

Note

APIs WebSocket é compatível somente com TLS 1.2. Versões anteriores do TLS não são compatíveis.

Criar uma API WebSocket usando comandos da AWS CLI

Para criar uma API WebSocket usando a AWS CLI, é necessário chamar o comando `create-api`, conforme mostrado no exemplo a seguir, que cria uma API com a expressão de seleção de rotas `$request.body.action`:

```
aws apigatewayv2 --region us-east-1 create-api --name "myWebSocketApi3" --protocol-type WEBSOCKET --route-selection-expression '$request.body.action'
```

Exemplos de resultado:

```
{  
    "ApiKeySelectionExpression": "$request.header.x-api-key",  
    "Name": "myWebSocketApi3",  
    "CreatedDate": "2018-11-15T06:23:51Z",  
    "ProtocolType": "WEBSOCKET",  
    "RouteSelectionExpression": "'$request.body.action'",  
    "ApiId": "aabbcdddee"  
}
```

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.

Para criar uma API WebSocket usando o console do API Gateway

1. Faça login no console do API Gateway e escolha Criar API.
2. Em Escolha o protocolo, selecione o WebSocket.
3. Em Criar nova API, escolha Nova API.
4. Em Configurações, no campo Nome da API, digite o nome da sua API, como **PetStore**, por exemplo.
5. Insira uma Expressão de seleção de rotas para sua API, como `$request.body.action`, por exemplo.

Para obter mais informações sobre expressões de seleção de rota, consulte [the section called “” \(p. 652\)](#).

6. Se desejar, digite uma Descrição para sua API.
7. Escolha Create API (Criar API).

Configurar rotas para uma API WebSocket no API Gateway

Tópicos

- [Configurar rotas para uma API WebSocket no API Gateway \(p. 635\)](#)
- [Especifique as configurações de solicitação de rota \(p. 636\)](#)

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 desejado, você pode

criar rotas personalizadas. Para obter mais informações, consulte [the section called “Sobre APIs WebSocket” \(p. 629\)](#).

Note

Na CLI, você pode criar rotas antes ou depois de gerar integrações, sendo possível reutilizar a mesma integração para várias rotas.

Tópicos

- [Criar uma rota usando o console do API Gateway \(p. 636\)](#)
- [Criar uma rota usando a AWS CLI \(p. 636\)](#)

Criar uma rota usando o console do API Gateway

Para criar uma rota usando o console do API Gateway

1. Faça login no console do API Gateway, escolha a API e selecione Rotas.
2. Para criar uma das rotas predefinidas (\$connect, \$disconnect e \$default), escolha seu nome.
3. Se desejado, você pode criar rotas personalizadas. Para fazer isso, insira o nome da chave de rota na caixa de texto Nova chave de rota e escolha o ícone de marca de seleção.

Note

Ao criar uma rota personalizada, não use o prefixo \$ no nome da chave de rota. Este prefixo está reservado para rotas predefinidas.

Criar uma rota usando a AWS CLI

Para criar uma rota usando a AWS CLI, chame `create-route`, conforme mostrado no exemplo a seguir.

```
aws apigatewayv2 --region us-east-1 create-route --api-id aabbccdde --route-key $default
```

Exemplos de resultado:

```
{  
    "ApiKeyRequired": false,  
    "AuthorizationType": "NONE",  
    "RouteKey": "$default",  
    "RouteId": "1122334"  
}
```

Especifique as configurações de solicitação de rota

Especifique 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” \(p. 630\)](#).

- Autorização: Se a autorização não for necessária, você pode especificar `NONE`. Caso contrário, especifique:
 - `AWS_IAM` para utilizar políticas AWS padrão do 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 autorizadores do Lambda do API Gateway \(p. 396\)](#).

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 usando o console” \(p. 403\)](#).

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: Você pode, opcionalmente, exigir uma chave de API para uma rota \$connect de 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 “Utilização de planos de uso com chaves de API” \(p. 450\)](#).

Configurar uma 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. Faça login no console do API Gateway, escolha a API e selecione Rotas.
2. Em Rotas, escolha \$connect.
3. Selecione Solicitação de rota no painel da visão geral de rotas.
4. Em Configurações de acesso, defina as configurações de rota da seguinte forma:
 - a. Para editar a configuração de Autorização, selecione o ícone de lápis. Escolha a configuração desejada no menu suspenso e selecione o ícone de marca de seleção para salvar a nova configuração.
 - b. Para editar a configuração de Chave de API obrigatória, selecione o ícone de lápis. Escolha true ou false no menu suspenso e selecione o ícone de marca de seleção para salvar a nova configuração.

Configurar integrações da API WebSocket no API Gateway

Tópicos

- [Configurar uma solicitação de integração da API WebSocket no API Gateway \(p. 637\)](#)
- [Configurar respostas de integração da API WebSocket no API Gateway \(p. 640\)](#)

Configurar uma solicitação de integração da API WebSocket no API Gateway

A configuração de uma solicitação de integração envolve o seguinte:

- Escolher uma chave de roteamento a ser integrada ao back-end.
- Especificar o endpoint de back-end para invocar, como um serviço da AWS ou endpoint HTTP.
- 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

Para adicionar uma solicitação de integração para uma rota em uma API WebSocket usando o console do API Gateway

1. Faça login no console do API Gateway, escolha a API e selecione Rotas.
2. Em Rotas, selecione a rota.
3. No painel de Visão geral de rotas, selecione Solicitação de integração.
4. Para Tipo de integração, escolha uma das seguintes opções:
 - Escolha Função do Lambda somente se a sua API for integrada a uma função do AWS Lambda que você já criou nesta conta ou em outra conta.

Para criar uma nova função do Lambda no AWS Lambda, definir uma permissão de recursos na função do Lambda ou executar qualquer outra ação de serviço do Lambda, opte por selecionar Serviço da AWS.

- Escolha HTTP se a sua API for integrada a um endpoint HTTP existente. Para obter mais informações, consulte [Configurar integrações HTTP no API Gateway \(p. 247\)](#).
 - Escolha Simular se deseja gerar respostas de API diretamente do API Gateway, sem a necessidade de um back-end de integração. Para obter mais informações, consulte [Configurar integrações simuladas no API Gateway \(p. 260\)](#).
 - Escolha Serviço da AWS; se sua API for integrada a um serviço da AWS.
 - Escolha Link do VPC se a sua API utilizar VpcLink como um endpoint de integração privada. Para obter mais informações, consulte [Configurar integrações privadas do API Gateway \(p. 252\)](#).
5. Se você escolheu Lambda Function (Função do Lambda), faça o seguinte:
 - a. Para Lambda Region (Região do Lambda), escolha o identificador de região que corresponde ao nome da região na qual você criou a função do Lambda. Por exemplo, se você tivesse criado a função do Lambda na região Leste dos EUA (Norte da Virgínia), escolheria **us-east-1**. Para obter uma lista de nomes e identificadores de região, consulte [AWS Lambda](#) na Referência geral do Amazon Web Services.
 - b. Para usar a integração de proxy do Lambda, escolha a caixa de seleção se você pretende usar a [integração de proxy do Lambda \(p. 227\)](#) ou a [integração de proxy do Lambda entre contas \(p. 34\)](#).
 - c. Para a função do Lambda, especifique a função de uma das seguintes formas:
 - Se a sua função do Lambda estiver na mesma conta, comece a digitar o nome da função e, em seguida, escolha a função 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, digite o ARN para a função.
- d. Selecione Invocar com credenciais do chamador se você deseja que o API Gateway invoque a função do Lambda usando as credenciais recebidas na solicitação de entrada.
- e. Para Função de execução, digite o ARN da função de invocação do Lambda que permite que o API Gateway invoque funções do Lambda.

- f. Para usar o valor de tempo limite padrão de 29 segundos, deixe a opção Usar tempo limite padrão desmarcada. Para definir um tempo limite personalizado, desmarque a caixa de seleção e insira um valor de tempo limite entre 50 e 29000 milissegundos.
 - g. Escolha Salvar.
6. Se você escolher HTTP, siga as instruções na etapa 4 de [the section called “Configurar uma solicitação de integração usando o console” \(p. 223\)](#).
 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” \(p. 223\)](#).
 9. Se você escolheu Link do VPC, faça o seguinte:
 - a. Para Usar a integração de proxy, escolha a caixa de seleção se você deseja que as solicitações sejam transmitidas para o endpoint do VPCLink.
 - b. Na lista suspensa de Link do VPC, escolha [Use Stage Variables] e digite \${stageVariables.vpcLinkId} na caixa de texto abaixo da lista.

Definiremos a variável do estágio vpcLinkId após implantar a API em um estágio e definir seu valor para o ID do VpcLink.
 - c. Em HTTP method (Método HTTP), escolha o tipo de método HTTP mais parecido com o método no back-end HTTP.
 - d. Para URL de endpoint, digite a URL do back-end HTTP que você deseja que essa integração use.
 - e. Para usar o valor de tempo limite padrão de 29 segundos, deixe a opção Usar tempo limite padrão desmarcada. Para definir um tempo limite personalizado, desmarque a caixa de seleção e insira um valor de tempo limite entre 50 e 29000 milissegundos.
10. Em Modelos de solicitação, faça o seguinte:
 - Para Expressão de seleção de modelo, escolha o ícone de lápis e substitua a palavra template por uma expressão de seleção de modelo. Esta é uma expressão que o API Gateway procura na carga da mensagem. 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.

Para obter mais informações sobre expressões de seleção de modelo, consulte [the section called “” \(p. 654\)](#).

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. Execute o comando `create-integration` conforme mostrado no exemplo a seguir:

```
aws apigatewayv2 --region us-east-1 create-integration --api-id aabbccdde --cli-input-json file://integration-params.json
```

A seguir está a saída de amostra para este exemplo:

```
{  
    "PassthroughBehavior": "WHEN_NO_MATCH",  
    "TimeoutInMillis": 29000,  
    "ConnectionType": "INTERNET",  
    "IntegrationResponseSelectionExpression": "${response.statuscode}",  
    "RequestTemplates": {  
        "application/json": "{\"statusCode\":200}"  
    },  
    "IntegrationId": "0abcdef",  
    "IntegrationType": "MOCK"  
}
```

Como alternativa, você pode configurar uma solicitação de integração para uma integração de proxy usando a AWS CLI, conforme mostrado no exemplo a seguir:

1. Crie uma função do Lambda no console do Lambda e forneça uma função de execução básica do Lambda.
2. Execute o comando [create-integration](#), conforme mostrado no exemplo a seguir:

```
aws apigatewayv2 create-integration --api-id aabbccdde --integration-type  
AWS_PROXY --integration-method POST --integration-uri arn:aws:apigateway:us-  
east-1:lambda:path/2015-03-31/functions/arn:aws:lambda:us-  
east-1:123412341234:function:simpleproxy-echo-e2e/invocations
```

A seguir está a saída de amostra para este exemplo:

```
{  
    "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"  
}
```

Configurar respostas de integração da API WebSocket no API Gateway

Tópicos

- [Visão geral das respostas de integração \(p. 640\)](#)
- [Configurar uma resposta de integração usando o console do API Gateway \(p. 641\)](#)
- [Configurar uma resposta de integração usando a AWS CLI \(p. 642\)](#)

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 back-end. Existem algumas diferenças na configuração de uma resposta de integração entre uma API REST e uma API WebSocket, que são descritas abaixo, mas, em termos de conceito, o comportamento é o mesmo.

As rotas do WebSocket podem ser configuradas para comunicação bidirecional ou unidirecional. Se uma rota apresentar uma resposta de rota, será configurada para comunicação bidirecional. Caso contrário, será configurada para comunicação unidirecional.

- Quando uma rota está configurada para comunicação bidirecional, uma resposta de integração permite que você configure as transformações na carga 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 restante deste documento pressupõe que você tenha optado por configurar uma rota com 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 back-end para o chamador 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 back-end. 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 as limitações atuais de expressões de seleção de modelo, consulte [the section called “” \(p. 654\)](#).

Configurar uma resposta de integração usando o console do API Gateway

Para 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, escolha a API e selecione Rotas.
2. Escolha a rota.
3. Escolha Resposta de integração.
4. Em Respostas de integração, insira um valor na caixa de texto Expressão de seleção de resposta.
5. Em Chave de resposta, selecione Adicionar resposta.

- a. Para definir uma chave de resposta de integração, insira um nome de chave na caixa de texto Nova chave de resposta e selecione o ícone de marca de seleção.
- b. Escolha o ícone de lápis ao lado da Expressão de seleção de modelo e insira uma expressão para que o API Gateway realize uma busca em sua mensagem de saída. Essa expressão deve ser avaliada como um valor de chave de resposta de integração que mapeia para um dos seus modelos de resposta.

Para obter mais informações sobre expressões de seleção de modelo, consulte [the section called “” \(p. 654\)](#).

Configurar uma resposta de integração usando a AWS CLI

Para configurar uma resposta de integração para uma API WebSocket usando a AWS CLI, chame o comando `create-integration-response`. O comando da CLI a seguir mostra um exemplo de configuração de uma resposta de 200:

```
aws apigatewayv2 create-integration-response \
--api-id vaz7da96z6 \
--integration-response-key 200
```

Configurar respostas de rota para uma API WebSocket no API Gateway

As rotas do WebSocket podem ser configuradas para comunicação bidirecional ou unidirecional. Se uma rota apresentar uma resposta de rota, será configurada para comunicação bidirecional. Caso contrário, será configurada para comunicação unidirecional. As respostas de rota são utilizadas para permitir comunicações bidirecionais em que sua API pode enviar uma resposta de volta ao cliente no contexto da mensagem do cliente.

Você pode configurar as respostas de rota e expressões de seleção de resposta ao utilizar o console do API Gateway, a AWS CLI ou um SDK da AWS. Para obter mais informações sobre respostas de rota, consulte [the section called “Como invocar a integração de seu back-end” \(p. 631\)](#).

Para obter mais informações sobre expressões de seleção de respostas de rota, consulte [the section called “” \(p. 654\)](#).

Tópicos

- [Configurar uma resposta de rota usando o console do API Gateway \(p. 642\)](#)
- [Configurar uma resposta de rota usando o console da AWS CLI \(p. 643\)](#)

Configurar uma resposta de rota usando o console do API Gateway

Para configurar uma resposta de rota para uma API WebSocket usando o console do API Gateway:

1. Faça login no console do API Gateway e selecione a API.
2. Em Rotas, selecione a rota.
3. Selecione Resposta de rota no painel de visão geral de rotas.

4. Em Modelagem de resposta, na caixa Expressão de seleção de resposta, insira a expressão de seleção de resposta desejada e escolha o ícone de marca de seleção.
5. Em Respostas de rota, em Chave de resposta, selecione Adicionar resposta.

Note

Atualmente, somente `$default` é compatível com as respostas de rota para APIs WebSocket.

6. Insira o nome da chave de resposta e escolha o ícone de marca de seleção.

Configurar uma resposta de rota usando o console da AWS CLI

Para configurar uma resposta de rota para uma API WebSocket usando a AWS CLI, chame o comando `create-route-response`, conforme mostrado no exemplo a seguir.

```
aws apigatewayv2 --region us-east-1 create-route-response --api-id aabbccdde --route-id 1122334 --route-response-key $default
```

Exemplos de resultado:

```
{  
    "RouteResponseId": "abcdef",  
    "RouteResponseKey": "$default"  
}
```

Implantar uma API 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 \(p. 6\)](#) e a associa a um [estágio \(p. 6\)](#). Cada estágio é um snapshot da API, sendo disponibilizado para ser chamado pelos aplicativos do cliente.

Important

Toda vez que atualiza uma API, o que inclui a modificação de rotas, métodos, integrações, autorizadores e qualquer outra atividade diferente das configurações de estágio, você deve reimplantar a API em um estágio novo ou já existente.

Por padrão, você está limitado a 10 estágios por API, portanto, é recomendável reutilizá-los.

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 no tamanho máximo de 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 “Configurar um nome de domínio personalizado de API” \(p. 676\)](#).

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 reimplementação da API ao estágio `prod`.

Tópicos

- [Criar uma implantação de API WebSocket usando a AWS CLI \(p. 644\)](#)
- [Criar uma implantação de API WebSocket usando o console do API Gateway \(p. 645\)](#)

Criar uma implantação de API WebSocket usando a AWS CLI

Para utilizar a AWS CLI visando criar uma implantação, utilize o comando `create-deployment` conforme mostrado no exemplo a seguir:

```
aws apigatewayv2 --region us-east-1 create-deployment --api-id aabbcccddee
```

Exemplos de resultado:

```
{  
    "DeploymentId": "fedcba",  
    "DeploymentStatus": "DEPLOYED",  
    "CreatedDate": "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.

Para criar um novo estágio e associá-lo à implantação, utilize o comando `create-stage` conforme mostrado no exemplo a seguir:

```
aws apigatewayv2 --region us-east-1 create-stage --api-id aabbcccddee --deployment-id fedcba  
--stage-name test
```

Exemplos de resultado:

```
{  
    "StageName": "test",  
    "CreatedDate": "2018-11-15T06:50:28Z",  
    "DeploymentId": "fedcba",  
    "DefaultRouteSettings": {  
        "MetricsEnabled": false,  
        "ThrottlingBurstLimit": 5000,  
        "DataTraceEnabled": false,  
        "ThrottlingRateLimit": 10000.0  
    },  
}
```

```
"LastUpdatedDate": "2018-11-15T06:50:28Z",  
"StageVariables": {},  
"RouteSettings": {}  
}
```

Para reutilizar um estágio já existente, atualize a propriedade `deploymentId` do estágio com o ID de implantação recém-criado (`{deployment-id}`) ao utilizar o comando `update-stage`.

```
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 visando criar uma implantação para uma API WebSocket:

1. Faça login no console do API Gateway e escolha a API.
2. No menu suspenso Ações, escolha Implantar API.
3. Escolha o estágio desejado na lista suspensa ou insira o nome de um novo estágio.

Invocar uma API WebSocket

Depois que você tiver implantado a API WebSocket, os aplicativos do cliente podem se conectar a ela para enviar mensagens, e back-end pode enviar mensagens para aplicativos de clientes 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 WebSocket e enviar mensagens a ela” \(p. 645\)](#).
- Você pode usar a API `@connections` do seu serviço de back-end para enviar uma mensagem de retorno a um cliente conectado, obter informações de conexão ou desconectar o cliente. Consulte [the section called “Use os comandos @connections em seu serviço de back-end” \(p. 646\)](#).
- Um aplicativo do cliente pode usar sua própria biblioteca WebSocket para invocar a API WebSocket.

Use wscat para se conectar a uma API WebSocket e enviar mensagens a ela

O utilitário `wscat` é uma ferramenta conveniente para testar uma API WebSocket que você tiver criado e implantado 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 ao executar 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://aabbcdddee.execute-api.us-east-1.amazonaws.com/test/
```

Você precisará substituir `aabbccdde` pelo ID da 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 sua API digite 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 se desconectar de sua API, digite `ctrl-C`.

Use os comandos @connections em seu serviço de back-end

Seu serviço de back-end 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 \(p. 647\)](#), portanto, você deve assiná-las com o [Signature versão 4 \(SigV4\)](#). Para fazer isso, é possível usar a API de gerenciamento do API Gateway. Para obter mais informações, consulte [ApiGatewayManagementApi](#).

No comando a seguir, você precisará substituir `{api-id}` pelo ID da 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.

Para enviar uma mensagem de retorno ao cliente, utilize:

```
POST https://{api-id}.execute-api.us-east-1.amazonaws.com/{stage}/@connections/{connection_id}
```

Você pode 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 precisará codificar o comando na URL, 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}
```

Note

O Postman não codifica o URL @connections, portanto, será necessário substituir os caracteres = e @ no URL codificado.

Você pode criar dinamicamente uma URL de retorno de chamada ao utilizar as variáveis \$context em sua integração. Por exemplo, se você usar a integração de proxy do Lambda com uma função Node.js do Lambda, é possível criar a URL da seguinte forma:

```
exports.handler = function(event, context, callback) {  
    var domain = event.requestContext.domain;  
    var stage = event.requestContext.stage;  
    var connectionId = event.requestContext.connectionId;  
    var callbackUrl = util.format(util.format('https://%s/%s/@connections/%s', domain, stage,  
        connectionId));  
    // Do a SigV4 and then make the call  
}
```

Controlar o acesso à API WebSocket no API Gateway

Tópicos

- [Utilizar autorização do IAM \(p. 647\)](#)
- [Criar uma função do autorizador REQUEST do Lambda \(p. 648\)](#)

Utilizar autorização do IAM

A autorização do IAM em APIs WebSocket é semelhante à utilizada para [APIs REST \(p. 384\)](#), 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
```

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.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "execute-api:Invoke"  
            ],  
            "Resource": [  
                "arn:aws:execute-api:us-east-1:account-id:api-id/prod/*"  
            ]  
        },  
        {  
            "Effect": "Deny",  
            "Action": [  
                "execute-api:Invoke"  
            ],  
            "Resource": [  
                "arn:aws:execute-api:us-east-1:account-id:api-id/prod/secret"  
            ]  
        },  
        {  
            "Effect": "Deny",  
            "Action": [  
                "execute-api:ManageConnections"  
            ],  
            "Resource": [  
                "arn:aws:execute-api:us-east-1:account-id:api-id/*"  
            ]  
        }  
    ]  
}
```

Criar uma função do autorizador REQUEST do Lambda

Uma função do autorizador do Lambda em APIs WebSocket é semelhante à utilizada para [APIs REST \(p. 398\)](#), com as seguintes exceções:

- 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 de função do autorizador do Lambda a seguir é uma versão WebSocket da função do autorizador do Lambda para APIs REST em [the section called “Criar uma função do autorizador do Lambda no console do Lambda” \(p. 398\)](#):

```
exports.handler = function(event, context, callback) {  
    console.log('Received event:', JSON.stringify(event, null, 2));  
  
    // 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, QueryString1 query parameter,
```

```
// stage variable of StageVar1 and the accountId in the request context all match
// specified values of 'headerValue1', 'queryValue1', 'stageValue1', and
// '123456789012', respectively.

// 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 restApiId = 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 = {};
  conditionIpAddress = {};

if (headers.HeaderAuth1 === "headerValue1"
  && queryStringParameters.QueryString1 === "queryValue1"
  && stageVariables.StageVar1 === "stageValue1"
  && requestContext.accountId === "123456789012") {
  callback(null, generateAllow('me', event.methodArn));
} else {
  callback("Unauthorized");
}
}

// Help function to generate an IAM policy
var generatePolicy = function(principalId, effect, resource) {
  // Required output:
  var authResponse = {};
    authResponse.principalId = principalId;
  if (effect && resource) {
    var policyDocument = {};
      policyDocument.Version = '2012-10-17'; // default version
    policyDocument.Statement = [];
    var statementOne = {};
      statementOne.Action = 'execute-api:Invoke'; // default action
    statementOne.Effect = effect;
      statementOne.Resource = resource;
    policyDocument.Statement[0] = statementOne;
    authResponse.policyDocument = policyDocument;
  }
  // Optional output with custom properties of the String, Number or Boolean type.
  authResponse.context = {
    "stringKey": "stringval",
    "numberKey": 123,
    "booleanKey": true
  };
  return authResponse;
}

var generateAllow = function(principalId, resource) {
  return generatePolicy(principalId, 'Allow', resource);
}

var generateDeny = function(principalId, resource) {
  return generatePolicy(principalId, 'Deny', resource);
```

```
}
```

Para configurar a função anterior do Lambda como uma função do autorizador REQUEST para uma API WebSocket, siga o mesmo procedimento para [APIs REST \(p. 403\)](#).

Para configurar a rota \$connect visando utilizar esse autorizador do Lambda no console, selecione ou crie a rota \$connect. Escolha a solicitação de rota e selecione seu autorizador no menu suspenso Autorização.

Para testar o autorizador, você precisará criar uma nova conexão. A alteração do autorizador em \$connect não afeta o cliente já conectado. Quando você se conectar à sua API WebSocket, será necessário fornecer valores para 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?QueryAuth1=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
```

Monitorar a execução da API WebSocket com CloudWatch

Você pode usar as métricas e logs do [Amazon CloudWatch](#) para monitorar a execução de APIs WebSocket. A configuração é semelhante à usada para APIs REST.

Para obter instruções sobre como configurar o registro de acesso e execução do CloudWatch, consulte [the section called “Configurar o registro de API do CloudWatch em logs usando o console do API Gateway” \(p. 539\)](#).

Quando você especifica o Formato de registro, é possível escolher em quais variáveis de contexto se registrar. Aqui está o exemplo de uma lista em formato JSON de variáveis de contexto para uma API WebSocket:

```
{
    "apiId" : "$context.apiId",
    "routeKey" : "$context.routeKey",
    "authorizer" : "$context.authorizer",
    "messageId" : "$context.messageId",
    "integrationLatency" : "$context.integrationLatency",
    "eventType" : "$context.eventType",
    "error" : "$context.error",
    "extendedRequestId" : "$context.extendedRequestId",
    "requestTime" : "$context.requestTime",
    "stage" : "$context.stage",
    "connectedAt" : "$context.connectedAt",
    "requestTimeEpoch" : "$context.requestTimeEpoch",
    "requestId" : "$context.requestId",
    "connectionId" : "$context.connectionId"
}
```

Você pode encontrar variáveis de contexto específicas a WebSocket aqui: [the section called “Referência de modelos de mapeamento WebSocket” \(p. 659\)](#)

As métricas a seguir são compatíveis com APIs WebSocket:

Métrica	Descrição
ConnectCount	Número de mensagens enviadas à integração de rotas \$connect.
MessageCount	O número de mensagens enviadas à API WebSocket, de ou para o cliente.
IntegrationError	Número de solicitações que retornam uma resposta 4XX/5XX da integração.
ClientError	Número de solicitações que têm uma resposta 4XX retornada pela API Gateway antes da integração ser invocada.
ExecutionError	Erros que ocorreram durante a chamada da integração.
IntegrationLatency	A diferença de hora entre o envio de solicitação para integração por parte do API Gateway e recebimento de resposta da integração por parte do API Gateway. Supressão para retornos de chamada e integrações simuladas.

Você pode usar as dimensões na tabela a seguir para filtrar métricas do API Gateway.

Dimensão	Descrição
Apild	Filtre métricas do API Gateway para uma API com o ID da API especificada.
ID da API, estágio	Filtre métricas do API Gateway para um estágio de API com o ID da API especificada e o ID do estágio.
ID da API, estágio, rota	Filtre métricas do API Gateway para um método de API com o ID da API especificada, ID do estágio e ID da rota. O API Gateway não enviará essas métricas, a menos que você habilite explicitamente as métricas detalhadas do CloudWatch. Isso pode ser feito ao chamar a ação UpdateStage da API REST V2 do API Gateway para atualizar a propriedade metricsEnabled como verdadeira.

Dimensão	Descrição
	Permitir essas métricas incorrerá em cobranças adicionais na conta. Para obter informações sobre a definição de preço, consulte Definição de preço do Amazon CloudWatch .

Expressões de seleção do WebSocket no API Gateway

Tópicos

- [Expressões de seleção de rota \(p. 652\)](#)
- [Expressões de seleção de modelo \(p. 654\)](#)
- [Expressões de seleção de modelo \(p. 654\)](#)
- [Expressões de seleção resposta de rotas \(p. 654\)](#)
- [Expressões de seleção de chave da API \(p. 654\)](#)
- [Expressões de seleção de mapeamento da API \(p. 654\)](#)
- [Expressões de seleção resposta da integração \(p. 654\)](#)
- [Resumo da expressão de seleção do WebSocket \(p. 655\)](#)

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 utilizar 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 discutir 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 cífrão (\$) 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.

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 usará a rota cuja routeKey corresponde exatamente ao valor avaliado. Se nenhuma correspondência for encontrada e uma rota com a chave \$default existir, 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 acima.

Expressão	Resultado avaliado	Descrição
<code>\$request.body.action</code>	<code>join</code>	Uma variável desempacotada
<code> \${request.body.action}</code>	<code>join</code>	Uma variável empacotada
<code> \${request.body.service}/join</code> <code> \${request.body.action}</code>	<code>chat/join</code>	Várias variáveis com valores estáticos
<code> \${request.body.action}-</code> <code> \${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 será 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 e, em seguida, o API Gateway tentará encontrar a rota `$default`, se disponível. Se a rota `$default` não for definida, o API Gateway retornará um erro.

Expressões de seleção de modelo

Ao definir uma [rota \(p. 636\)](#) para uma API WebSocket, você pode especificar 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 uma rota `requestmodels`.

Um modelo é expresso como um [esquema JSON](#) e descreve a estrutura de dados do corpo da solicitação. A natureza das expressões 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 “Criar modelos e modelos de mapeamento” \(p. 273\)](#).

Expressões de seleção de modelo

Quando você define uma [solicitação de integração \(p. 637\)](#) ou [resposta de integração \(p. 640\)](#) para uma API WebSocket, é possível especificar uma expressão de seleção de modelo. 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 resposta de rotas

Uma [resposta de rota \(p. 642\)](#) é utilizada para modelar uma resposta do back-end 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 \(p. 6\)](#) 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}`.

Expressões de seleção resposta da integração

Quando você [define uma resposta de integração \(p. 640\)](#) 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 chamador sem modelagem ou modificação.

Ao contrário de outras expressões de seleção descritas acima, 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 baseada 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 chamador.

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	Avaliar chave para	Observações	Exemplo de caso de uso
<code>Api.RouteSelectionForRequestKey</code>		\$default é compatível com uma rota genérica.	Roteie mensagens WebSocket com base no contexto de uma solicitação do cliente.
<code>Route.ModelSelectionForRequestModels</code>		Opcional. Se fornecida para integração não proxy, a validação do modelo ocorre.	Execute a validação de solicitação dinamicamente na mesma rota.
<code>Integration.TemporaryTemplateExpressionForRequestTemplates</code>		\$default é compatível como um genérico.	Manipular a

Expressão de seleção	Avaliar chave para	Observações	Exemplo de caso de uso
		<p>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>	<p>solicitação do chamador nas propriedades dinâmicas da solicitação.</p>

Expressão de seleção	Avaliar chave para	Observações	Exemplo de caso de uso
<code>Integration.IntegrationResponse.selectIntegrationResponseKey</code>		Opcional. Pode ser fornecida para integração não proxy. Funciona como uma correspondência de padrão para mensagens de erro (do Lambda ou códigos de status (de integrações HTTP)). \$default é necessário para integrações não proxy visando atuar como um genérico para respostas bem-sucedidas.	Manipular a resposta do back-end. Escolha a ação que deve ocorrer com determinada resposta dinâmica do back-end (por exemplo, manipular determinados erros de forma distinta).

Expressão de seleção	Avaliar chave para	Observações	Exemplo de caso de uso
<code>IntegrationResponse#keyPath</code>	IntegrationResponse#keyPath	Opcional. Pode ser fornecida para integração não proxy. O padrão \$ é compatível com a mesma rota e integração associada.	Em alguns casos, uma propriedade dinâmica da resposta pode ditar transformações diferentes para a mesma rota e integração associada. \${request.body.\$} \${integration.response.\$} \${integration.response.\$} \${integration.response.\$} e valores estáticos são compatíveis. \$default é compatível como um genérico.

Expressão de seleção	Avaliar chave para	Observações	Exemplo de caso de uso
<code>Route.RouteResponse\$RouteResponseExpression\$RouteResponseKey</code>		Deve ser fornecido para iniciar a comunicação bidirecional para uma rota do WebSocket. Atualmente, este valor é restrito apenas ao <code>\$default</code> .	
<code>RouteResponse.MaybePathExpression\$RequestModels</code>		Atualmente incompatível.	

Referência de modelos de mapeamento da API 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 Epoch .
<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: CONNECT, MESSAGE ou DISCONNECT.
<code>\$context.messageId</code>	Um ID exclusivo do servidor para uma mensagem. Disponível apenas quando o <code>\$context.eventType</code> é MESSAGE.
<code>\$context.routeKey</code>	A chave de roteamento selecionada.
<code>\$context.requestId</code>	Igual a <code>\$context.extendedRequestId</code> .

Parâmetro	Descrição
<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 a partir de uma função do Lambda do autorizador do Lambda do API Gateway (anteriormente conhecido como autorizador personalizado).
<code>\$context.authorizer.property</code>	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> <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.message</code>	Uma string de uma mensagem de erro do API Gateway. Essa variável pode ser utilizada apenas no registro de acesso.
<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. Essa variável pode ser utilizada apenas no registro de acesso.
<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>	O provedor de autenticação do Amazon Cognito usado pelo agente de chamada que está fazendo a solicitação. Disponível apenas se a solicitação tiver sido assinada com as credenciais do Amazon Cognito.
	Para obter informações relacionadas a esta e outras variáveis <code>\$context</code> do Amazon Cognito, consulte Uso de identidades federadas no Guia do desenvolvedor do Amazon Cognito.
<code>\$context.identity.sourceIp</code>	O endereço IP de origem da conexão TCP que está fazendo a solicitação ao 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 agente de chamada da API.
<code>\$context.identity.userArn</code>	O Nome do Recurso Amazon (ARN) do usuário efetivo identificado após a autenticação.
<code>\$context.integrationLatency</code>	A latência de integração em ms, disponível somente para registro de log de acesso.
<code>\$context.requestTime</code>	O horário da solicitação CLF formatado (dd/MMM/yyyy:HH:mm:ss +-hhmm).
<code>\$context.requestTimeEpoch</code>	O horário da solicitação Epoch formatado.
<code>\$context.stage</code>	O estágio de implantação da chamada de API (por exemplo, Beta ou Prod).
<code>\$input.body</code>	Retorna a carga bruta como uma string.
<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 JSONPath ou JSONPath para Java.</p>

Parâmetro	Descrição
<code>\$input.path(x)</code>	<p>Usa uma string de expressão JSONPath (<code>x</code>) e retorna uma representação de objeto JSON do resultado. Isso permite que você acesse e manipule elementos da carga nativamente em Apache VTL (Velocity Template Language).</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 JSONPath ou JSONPath para Java.</p>
<code>\$stageVariables.<variable_name></code>	<code><variable_name></code> representa um nome de variável de estágio.
<code>\$stageVariables['<variable_name>']</code>	<code><variable_name></code> representa qualquer nome de variável de estágio.
<code> \${stageVariables['<variable_name>']}</code>	<code><variable_name></code> representa qualquer nome de variável de estágio.

Parâmetro	Descrição
<code>\$util.escapeJavaScript()</code>	<p>Escapa os caracteres em uma string usando regras de string JavaScript.</p> <p>Note</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(data).replaceAll("\\", "")</pre>
<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'))) { "errorMessageObjKey2ArrVal" : \$errorMessageObj.key2.arr[0] }</pre> <p>Você receberá a seguinte saída:</p> <pre>{ "errorMessageObjKey2ArrVal" : 1 }</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.
<code>\$util.base64Decode()</code>	Decodifica os dados de uma string codificada em base64.

Publicar suas APIs do API Gateway

Assim que você implantar suas APIs, poderá publicá-las implantando um portal do desenvolvedor e vendê-las como SaaS por meio do AWS Marketplace. Também é possível criar um nome domínio personalizado para a sua API. Você pode exportar as definições de sua API para gerar um SDK para seus usuários chamarem a API usando uma linguagem de programação com suporte. E você pode usar uma [versão canary](#) para testar novas alterações.

Tópicos

- [Usar o Portal do desenvolvedor sem servidor para catalogar suas APIs do API Gateway \(p. 664\)](#)
- [Vender suas APIs do API Gateway pelo AWS Marketplace \(p. 673\)](#)
- [Configurar um nome de domínio personalizado para uma API no API Gateway \(p. 676\)](#)
- [Exportar uma API REST do API Gateway \(p. 700\)](#)
- [Definir uma implantação da versão Canary no API Gateway \(p. 704\)](#)

Usar o Portal do desenvolvedor sem servidor para catalogar suas APIs do API Gateway

Um portal do desenvolvedor é um aplicativo que você usa para disponibilizar suas APIs para os clientes. O Portal do desenvolvedor sem servidor pode ser usado para publicar APIs gerenciadas pelo API Gateway diretamente do API Gateway. Você também pode usá-lo para publicar APIs não gerenciadas pelo API Gateway ao fazer upload de definições de OpenAPI para elas. Quando você publica APIs em um Portal do desenvolvedor sem servidor, os clientes podem facilmente:

- Descobrir quais APIs estão disponíveis.
- Procurar a documentação da API.
- Registrar e receber imediatamente sua própria chave de API, que pode ser usada para criar aplicativos.
- Experimentar as suas APIs na interface do usuário do portal do desenvolvedor.
- Monitorar seu próprio uso da API.

O Amazon API Gateway publica um portal do desenvolvedor sem servidor atualizado regularmente no [AWS Serverless Application Repository](#) e [GitHub](#). Ele é lançado sob a [licença do Apache 2.0](#), permitindo sua personalização e incorporação às ferramentas de criação e implantação. O front-end é gravado no React e foi projetado para ser totalmente personalizável. Consulte <https://github.com/awslabs/aws-api-gateway-developer-portal/wiki/Customization>.

Para obter mais informações sobre o AWS Serverless Application Repository, consulte [Guia do desenvolvedor do AWS Serverless Application Repository](#).

Tip

Se você quiser testar um exemplo de portal do desenvolvedor sem servidor, consulte <https://developer.exampleapigateway.com/>.

Tópicos

- [Criar um portal do desenvolvedor \(p. 665\)](#)
- [Configurações do portal do desenvolvedor \(p. 666\)](#)
- [Criar um usuário administrador para seu portal de desenvolvedor \(p. 668\)](#)
- [Publicar uma API gerenciada pelo API Gateway no portal do desenvolvedor \(p. 668\)](#)

- [Atualizar ou excluir uma API gerenciada pelo API Gateway \(p. 669\)](#)
- [Para remover uma API não gerenciada pelo API Gateway \(p. 670\)](#)
- [Publicar uma API não gerenciada pelo API Gateway no portal do desenvolvedor \(p. 670\)](#)
- [Como seus clientes podem usar o portal do desenvolvedor \(p. 671\)](#)
- [Melhores práticas para portais do desenvolvedor \(p. 673\)](#)

Criar um portal do desenvolvedor

Você pode implantar o portal do desenvolvedor do API Gateway do modo como está ou personalizá-lo para se adequar à sua marca. Há duas maneiras de implantar o portal do desenvolvedor:

- Usando o AWS Serverless Application Repository. Basta selecionar o botão Deploy (Implantar) para iniciar a pilha do AWS CloudFormation do Portal do desenvolvedor sem servidor do API Gateway e inserir vários parâmetros de fila no console do Lambda.
- Fazendo download do portal de desenvolvedor sem servidor do API Gateway a partir do repositório GitHub da AWS e iniciando a pilha do AWS CloudFormation do portal do desenvolvedor sem servidor do API Gateway a partir da CLI do AWS SAM.

Para implantar o portal do desenvolvedor sem servidor usando o AWS Serverless Application Repository

1. Acesse a página do [portal do desenvolvedor sem servidor do API Gateway](#) no AWS Serverless Application Repository.
2. Selecione Deploy (Implantar).

Note

Pode ser solicitado que você efetue o login no console do AWS Lambda.

3. No console do Lambda, você precisará preencher os [parâmetros de pilha do portal do desenvolvedor \(p. 666\)](#) necessários em Application settings (Configurações do aplicativo).

Note

Os nomes de bucket do S3 e o prefixo de domínio do Amazon Cognito são necessários. As outras configurações são opcionais.

4. Se você deseja habilitar o botão de comentários de clientes Got an opinion? (Tem uma opinião?), você precisará:
 - Insira um endereço de e-mail na caixa DevPortalAdminEmail. Quando um cliente enviar comentários, um e-mail será enviado para esse endereço.

Note

Se você não fornecer um endereço de e-mail, Got an opinion? (Tem uma opinião?) não será exibido no portal do desenvolvedor.

- Você também pode inserir um nome de tabela na caixa DevPortalFeedbackTableName. O nome padrão é **DevPortalFeedback**. Quando um cliente enviar comentários, eles serão armazenados em uma tabela do DynamoDB com esse nome.
5. Marque a caixa de seleção ao lado de Eu reconheço que este aplicativo cria funções personalizadas do IAM.
 6. Selecione Deploy (Implantar).
 7. Se você inseriu um endereço de e-mail no parâmetro de pilha AdminEmail, um e-mail de assinatura do Amazon SNS é enviado para esse endereço de e-mail para confirmar que você deseja assinar o tópico do Amazon SNS que especificou na configuração MarketplaceSubscriptionTopicProductCode.

Para fazer download e implantar o portal do desenvolvedor sem servidor usando o AWS SAM

1. Verifique se você tem a [AWS CLI](#) mais recente e se a [CLI do AWS SAM](#) está instalada e configurada.
2. Faça download ou clone o repositório do [portal do desenvolvedor do API Gateway](#) em um diretório local.
3. Verifique se você tem um bucket do Amazon S3 acessível de forma privada para o qual fazer upload das funções Lambda compactadas. Se você não tiver, poderá criar um usando a CLI ou o console do Amazon S3.

Na CLI do SAM, execute o seguinte comando na raiz do projeto, substituindo `{your-lambda-artifacts-bucket-name}` pelo nome do bucket do Amazon S3:

```
sam package --template-file ./cloudformation/template.yaml --output-template-file ./cloudformation/packaged.yaml --s3-bucket {your-lambda-artifacts-bucket-name}
```

4. Para esta etapa, consulte [the section called “Configurações do portal do desenvolvedor” \(p. 666\)](#) para obter mais informações sobre parâmetros (como `CognitoDomainNameOrPrefix`) após `--parameter-overrides`.

Note

Verifique se você tem um bucket do Amazon S3 acessível de forma privada para fazer upload do modelo SAM. (O AWS CloudFormation lerá o modelo no bucket para implantar o portal do desenvolvedor.) Ele pode ser o mesmo bucket que você especificou na etapa anterior para fazer upload das funções Lambda compactadas.

Execute a linha de comando a seguir na raiz do projeto, substituindo:

- `{your-template-bucket-name}` pelo nome do bucket do Amazon S3.
- `{custom-prefix}` por um prefixo que é globalmente exclusivo
- `{cognito-domain-or-prefix}` por uma string exclusiva.

```
sam deploy --template-file ./cloudformation/packaged.yaml --s3-bucket {your-template-bucket-name} --stack-name "{custom-prefix}-dev-portal" --capabilities CAPABILITY_NAMED_IAM --parameter-overrides CognitoDomainNameOrPrefix= "{cognito-domain-or-prefix}" DevPortalSiteS3BucketName="{custom-prefix}-dev-portal-static-assets" ArtifactsS3BucketName="{custom-prefix}-dev-portal-artifacts"
```

Assim que o portal do desenvolvedor for totalmente implantado, você pode obter a URL conforme segue.

Para obter o URL do portal do desenvolvedor recém-criado

1. Abra o console de gerenciamento do AWS CloudFormation.
2. Escolha o nome da pilha (`aws-serverless-repository-api-gateway-dev-portal` é o nome padrão da pilha).
3. Abra a seção Outputs. O URL do portal do desenvolvedor é especificado na propriedade `WebSiteURL`.

Configurações do portal do desenvolvedor

Veja a seguir as configurações que você precisará definir durante e após a implantação do seu portal do desenvolvedor:

Note

Muitas das configurações a seguir poderão ser alteradas depois que você implantar seu portal do desenvolvedor, atualizando a pilha do AWS CloudFormation. Para obter mais informações, consulte [Atualizações de pilha do AWS CloudFormation](#).

ArtifactsS3BucketName (OBRIGATÓRIO)

O processo de implantação criará um bucket do Amazon S3 acessível de forma privada no qual os metadados do catálogo serão armazenados. Especifique um nome para atribuir ao bucket. Esse nome precisa ser globalmente exclusivo.

CognitoDomainNameOrPrefix (OBRIGATÓRIO)

Essa string é usada com a interface do usuário hospedada do Amazon Cognito para login e cadastro de usuários. Especifique uma string de prefixo ou nome de domínio único.

CognitoIdentityPoolName

O processo de implantação criará um grupo de identidades do Amazon Cognito. O nome padrão do grupo de identidades é DevPortalUserPool.

CustomDomainNameAcmCertArn

Se você forneceu um nome de domínio associado a um certificado do ACM, também deve especificar o ARN do certificado do ACM aqui. Deixe esse campo em branco para criar um portal do desenvolvedor sem um nome de domínio personalizado.

DevPortalAdminEmail

Especifique um endereço de e-mail para habilitar o botão de comentários de clientes Got an opinion? (Tem uma opinião?). Quando um cliente enviar comentários, um e-mail será enviado para esse endereço.

Note

Se você não fornecer um endereço de e-mail, o botão Got an opinion? (Tem uma opinião?) não será exibido no portal do desenvolvedor.

DevPortalFeedbackTableName

Se você especificar um endereço de e-mail para DevPortalAdminEmail, o processo de implantação criará uma tabela do DynamoDB para armazenar comentários inseridos por clientes. O nome padrão é **DevPortalFeedback**. Você também pode especificar seu próprio nome de tabela.

DevPortalCustomersTableName

O processo de implantação criará uma tabela do DynamoDB onde as contas de clientes serão armazenadas. Você também pode especificar um nome para atribuir a essa tabela. Ela deve ser exclusiva na região de sua conta. O nome padrão da tabela é DevPortalCustomers.

DevPortalSiteS3BucketName (OBRIGATÓRIO)

O processo de implantação criará um bucket do Amazon S3 acessível de forma privada no qual o código do aplicativo web será armazenado. Especifique um nome para atribuir a este bucket. Esse nome precisa ser globalmente exclusivo.

MarketplaceSubscriptionTopicProductCode

Esse é o sufixo do tópico do Amazon SNS para eventos de assinatura/cancelamento de assinatura. Insira o valor desejado. O valor padrão é **DevPortalMarketplaceSubscriptionTopic**. Essa configuração só é relevante se você estiver usando a integração do AWS Marketplace. Para obter mais informações, consulte [Configurar seu produto de SaaS para aceitar novos clientes](#).

StaticAssetRebuildMode

Por padrão, uma recompilação de ativo estático não substitui o conteúdo personalizado. Especifique **overwrite-content** para substituir o conteúdo personalizado por sua versão local.

Important

Se você especificar **overwrite-content**, todas as alterações personalizadas em seu bucket do Amazon S3 serão perdidas. Não faça isso a menos que você saiba o que está fazendo.

StaticAssetRebuildToken

Forneça um token diferente da última implantação para fazer upload novamente dos ativos estáticos do portal do desenvolvedor do site. Você pode fornecer um timestamp ou GUID em cada implantação para sempre fazer upload dos ativos novamente.

UseRoute53Nameservers

Aplicável somente se você estiver criando um nome de domínio personalizado para seu portal do desenvolvedor. Especifique **true** para ignorar a criação de um HostedZone e RecordSet do Route 53. Você precisará fornecer seu próprio servidor de nome que hospeda no lugar do Route 53. Caso contrário, deixe esse campo definido como **false**.

Criar um usuário administrador para seu portal de desenvolvedor

Assim que você tiver implantado seu portal do desenvolvedor, crie pelo menos um usuário administrador. Você pode fazer isso criando um novo usuário e o adicionando ao grupo de administradores para o grupo de usuários do Amazon Cognito que o AWS CloudFormation criou quando você implantou o portal do desenvolvedor.

Criar um usuário administrador

1. No seu portal do desenvolvedor, selecione Register (Registrar).
2. Insira um endereço de e-mail e uma senha e selecione Register (Registrar).
3. Em uma guia separada no navegador, entre no console do Amazon Cognito.
4. Selecione Manage User Pools.
5. Escolha o grupo de usuários para seu portal do desenvolvedor que você definiu quando implantou o portal do desenvolvedor.
6. Escolha o usuário para promover a administrador.
7. Selecione Add to group (Adicionar a grupo).
8. No menu suspenso, selecione **{stackname}-dev-portalAdminsGroup**, em que **{stackname}** é o nome da pilha de quando você implantou o portal do desenvolvedor.
9. Selecione Add to group (Adicionar a grupo).
10. Em seu portal do desenvolvedor, faça logout e login novamente usando as mesmas credenciais. Agora, você deve ver um link do Admin Panel (Painel do administrador) no canto superior direito, ao lado de My Dashboard (Meu painel).

Publicar uma API gerenciada pelo API Gateway no portal do desenvolvedor

As etapas a seguir descrevem como você, como proprietário da API, publica uma API no portal do desenvolvedor para que seus clientes possam se inscrever nela.

Etapa 1: Disponibilizar uma API gerenciada pelo API Gateway para publicação

1. Se você ainda não tiver feito isso, [implante a API em um estágio \(p. 518\)](#).

2. Se você ainda não tiver feito isso, crie um [plano de uso \(p. 450\)](#) e o associe ao estágio de API para a API implantada.

Note

Você não precisa associar chaves de API ao plano de uso. O portal do desenvolvedor fará isso para você.

3. Faça login no Developer Portal (Portal do desenvolvedor) e acesse o Painel do administrador.
4. A API deve estar presente na lista de APIs do Admin Panel (Painel do administrador). Na lista de APIs, as APIs são agrupadas por plano de uso. O grupo Not Subscribable (Não podem ser inscritas) No Usage Plan (Sem plano de uso) lista as APIs que não estão associadas a um plano de uso.

Etapa 2: Tornar a API gerenciada pelo API Gateway visível no portal do desenvolvedor

1. Na lista de APIs do Admin Panel (Painel do administrador), marque o valor Displayed (Exibido) para sua API. Quando for carregado pela primeira vez, esse valor será definido como False (Falso).
2. Para tornar a API visível no portal do desenvolvedor, selecione o botão False (Falso). Seu valor será alterado para True (Verdadeiro), indicando que a API agora está visível.

Tip

Para tornar visíveis todas as APIs em um plano de uso, selecione o botão False (Falso) ou Partial (Parcial) na barra de cabeçalho para o plano de uso.

3. Navegue até o painel de APIs para ver o portal do desenvolvedor da forma como seus clientes o veem. Você deve ver a sua API listada na coluna de navegação à esquerda.

Se a API for implantada em um estágio que está associado a um plano de uso, você verá um botão Assinatura (Assinatura) para a API. Este botão faz com que a chave de API do cliente seja associada ao plano de uso ao qual a API está associada.

Agora que a API gerenciada pelo API Gateway é exibida, você pode habilitar a geração de SDKs para que seus clientes façam download de um SDK para ela.

Etapa 3: Habilitar a geração de SDKs

1. Na lista de APIs do Admin Panel (Painel do administrador), selecione o botão Disabled (Desabilitado). Seu valor será alterado para Enabled (Habilitado), indicando que a geração de SDKs agora é permitida.
2. Navegue até o painel de APIs e selecione a sua API na coluna de navegação à esquerda. Agora, você deve ver um botão Download SDK (Fazer download do SDK) para ela.

Se você escolher o botão Download SDK (Fazer download do SDK), você deve ver uma lista de tipos de SDKs dos quais você pode fazer download.

Atualizar ou excluir uma API gerenciada pelo API Gateway

Se você fizer alterações em sua API no API Gateway depois que a tiver publicado, precisará reimplantá-la.

Para atualizar uma API gerenciada pelo API Gateway

1. Faça as alterações desejadas na API no console do API Gateway, na AWS CLI ou em um SDK.
2. Implante a API novamente no mesmo estágio de antes.

3. No portal do desenvolvedor, na lista de APIs do Admin Panel (Painel do administrador), selecione Update (Atualizar). Isso atualiza a API que é exibida na interface do usuário do portal do desenvolvedor.

Note

O botão Download SDK (Fazer download do SDK) sempre obtém a última versão implantada da sua API, mesmo se você não atualizá-la no portal do desenvolvedor.

4. Navegue até o painel de APIs e selecione a sua API na coluna de navegação à esquerda. Você deve ver suas alterações.

Para interromper a exibição de uma API na lista de APIs do portal do desenvolvedor (sem revogar o acesso dos seus clientes a ela):

1. Na lista de APIs do Admin Panel (Painel do administrador), marque o valor Displayed (Exibido) para sua API. Se a API estiver visível, esse valor será definido como True (Verdadeiro).
2. Para tornar a API invisível, selecione o botão True (Verdadeiro). Seu valor será alterado para False (Falso), indicando que a API agora está invisível.
3. Navegue até o painel de APIs. Você não deve mais ver a sua API listada na coluna de navegação à esquerda.

Para revogar o acesso dos clientes a uma API sem excluí-la inteiramente, você deve executar uma das seguintes ações no [console do API Gateway \(p. 457\)](#), na [API REST ou na CLI do API Gateway \(p. 462\)](#):

- Remova as chaves de API do plano de uso.
- Remova o estágio de API do plano de uso.

Para remover uma API não gerenciada pelo API Gateway

Para interromper a exibição de uma API não gerenciada pelo API Gateway e remover o acesso de seus clientes a ela, selecione Delete (Excluir). Isso não exclui a API, mas excluirá seu arquivo de especificação do OpenAPI do portal do desenvolvedor.

Publicar uma API não gerenciada pelo API Gateway no portal do desenvolvedor

As etapas a seguir descrevem como você publica APIs não gerenciadas pelo API Gateway (ou "genéricas") para seus clientes. Você pode fazer upload de definições do OpenAPI 2.0 (Swagger) ou 3.x para as APIs em arquivos JSON, YAML ou YAML.

Para publicar uma API não gerenciada pelo API Gateway no portal do desenvolvedor

1. Faça login no Developer Portal (Portal do desenvolvedor) e acesse o Painel do administrador.
2. Em Generic APIs (APIs genéricas), selecione Add API (Adicionar API).
3. Navegue até o diretório onde os arquivos do OpenAPI para suas APIs residem e selecione os arquivos para upload.
4. Escolha Carregar.

Note

Se qualquer um dos arquivos não puder ser analisado ou não contiver um título de API, você receberá um aviso, e esses arquivos não serão carregados. Todos os outros arquivos serão carregados. Você precisará selecionar o ícone x para descartar a caixa de diálogo.

5. Agora, você deve ver a sua API listada em Generic APIs (APIs genéricas). Se não fizer isso, saia do Admin Panel (Painel do administrador) e volte para atualizar a lista.

Como seus clientes podem usar o portal do desenvolvedor

Para criar aplicativos e testar as APIs, seus clientes precisarão criar contas de desenvolvedor registrando-se em seu portal do desenvolvedor.

Para criar uma conta de desenvolvedor e obter uma chave de API

1. No portal do desenvolvedor, selecione Register (Registrar).
2. Insira um endereço de e-mail e uma senha e selecione Register (Registrar).
3. Para localizar a chave de API, selecione My Dashboard (Meu painel).

Uma conta de desenvolvedor fornece ao seu cliente uma chave de API, que normalmente é necessária para usar e testar as APIs, e permite o rastreamento de uso para você e seus clientes.

Quando o cliente fizer o primeiro registro, sua nova chave de API não estará vinculada a qualquer uma das suas APIs.

Para ativar a chave de API para uma API

1. Escolha APIs.
2. Escolha a API na lista de APIs e selecione Subscribe (Inscrever-se).

Isso faz com que a chave de API seja associada ao plano de uso ao qual a API está associada.

Agora o cliente se inscreveu na API e poderá fazer chamadas para os métodos na API. Estatísticas de uso diário da API serão exibidas em MyDashboard.

Um cliente pode testar os métodos de API na interface do usuário do portal do desenvolvedor sem a assinatura.

Note

Se qualquer um dos métodos de API exigir uma chave de API, um botão Authorize (Atualizar) será exibido acima da lista de métodos. Os métodos que exigem uma chave de API terão um ícone de cadeado preto. Para experimentar métodos que exigem uma chave de API, selecione o botão Authorize (Autorizar) e insira uma chave de API válida que está associada ao plano de uso para o estágio da API.

Você pode ignorar o botão Authorize (Autorizar) com segurança quando ele é exibido em APIs que você já está inscrito.

Para experimentar uma API sem assinatura

1. Navegue até a API na lista de APIs.

2. Selecione um método de API para expandi-lo.
3. Selecione Try it out (Experimentar).
4. Selecione Execute (Executar).

Seus clientes podem enviar comentários para você da seguinte forma:

Para enviar comentários para uma API

1. Selecione Got an opinion? (Tem uma opinião?).
2. Na janela pop-up, insira comentários.
3. Selecione Enviar.

Os comentários do cliente são armazenados na tabela do DynamoDB para o portal do desenvolvedor. O nome padrão dessa tabela é DevPortalFeedback. Além disso, um e-mail é enviado para os endereços de e-mail que foram especificados no campo DevPortalAdminEmail. Se nenhum endereço de e-mail foi especificado quando o portal do desenvolvedor foi implantado, Got an opinion? (Tem uma opinião?) não será exibido.

Note

Se você alterar o nome dessa tabela, use um nome que seja exclusivo na região da sua conta.

Se você tiver habilitado a geração de SDKs para a API no Admin Panel (Painel do administrador), o cliente poderá fazer download de um SDK para ela.

Para fazer download de um SDK para uma API

1. Selecione o botão Download SDK (Fazer download do SDK) para a API desejada.
2. Selecione a linguagem ou plataforma do SDK desejada da seguinte forma:
 - Para Android:
 1. Para Group ID, digite o identificador exclusivo para o projeto correspondente. Ele é usado no arquivo `pom.xml` (por exemplo, **com.mycompany**).
 2. Para Invoker package, digite o namespace para as classes de cliente geradas (por exemplo, **com.mycompany.clientsdk**).
 3. Para Artifact ID, digite o nome do arquivo .jar compilado sem a versão. Ele é usado no arquivo `pom.xml` (por exemplo, **aws-apigateway-api-sdk**).
 4. Em Artifact version, digite o número de versão do artefato para o cliente gerado. Ele é usado no arquivo `pom.xml` e deve seguir um padrão **principal.secundário.patch** (por exemplo, **1.0.0**).
 - Para JavaScript, o SDK é obtido por download imediatamente.
 - Para iOS (Objective-C) ou iOS (Swift), digite um prefixo exclusivo na caixa Prefixo (Prefixo).

O efeito do prefixo é o seguinte: se você atribuir, por exemplo, SIMPLE_CALC como o prefixo para o SDK da API [SimpleCalc \(p. 554\)](#) 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_CALCOuput 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 [Use o SDK do iOS gerado pelo API Gateway para uma API REST em Objective-C ou Swift \(p. 575\)](#).

- Em Java:

1. 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.
 2. 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.
- Para Ruby, em Service Name (Nome do serviço), 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.

Melhores práticas para portais do desenvolvedor

Veja a seguir as sugestões de melhores práticas a serem seguidas ao implantar um portal do desenvolvedor.

- Na maioria dos casos, implante apenas um portal do desenvolvedor para todas as suas APIs. Em alguns casos, você pode optar por ter portais do desenvolvedor separados para versões de produção e desenvolvimento de suas APIs. Não é recomendado usar mais de um portal do desenvolvedor para APIs de produção.
- Ao criar um plano de uso e associá-lo a um estágio, você não precisa associar as chaves de API ao plano de uso. O portal do desenvolvedor fará isso para você.
- Observe que, se APIs em um plano estiverem visíveis, todas as APIs nesse plano de uso poderão ser assinadas, mesmo se você não as tornou visíveis para seus clientes.

Vender suas APIs do API Gateway pelo AWS Marketplace

Depois de criar, testar e implantar suas APIs, será possível empacotá-las em um [plano de uso \(p. 450\)](#) do API Gateway e vender o plano como um produto de Software como Serviço (SaaS) pelo 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, é necessário configurar o canal de vendas para integrar o AWS Marketplace ao API Gateway. De um modo geral, isso envolve listar seu produto no AWS Marketplace, configurar uma função IAM com as políticas adequadas para permitir que o API Gateway envie métricas de uso ao 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 permitir que seus clientes comprem seu produto no AWS Marketplace, você deve registrar seu portal de desenvolvedor (um aplicativo externo) no AWS Marketplace. O portal do desenvolvedor deve lidar com as solicitações de assinatura que são redirecionadas do console do AWS Marketplace.

Para um exemplo de aplicativo de portal do desenvolvedor, consulte o [Portal do desenvolvedor do API Gateway](#).

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

- [Iniciar a integração do AWS Marketplace com o API Gateway \(p. 674\)](#)
- [Gerenciar a assinatura do cliente para planos de uso \(p. 675\)](#)

Iniciar a integração do AWS Marketplace com o API Gateway

As tarefas a seguir são para a inicialização única da integração do AWS Marketplace ao API Gateway, o que permite a venda da 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 IAM denominada `ApiGatewayMarketplaceMeteringRole` com a seguinte política de execução e política de confiança. Essa função permite que o API Gateway envie métricas de uso ao AWS Marketplace em seu nome.

Política de execução da função de medição

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Action": [  
                "aws-marketplace:BatchMeterUsage",  
                "aws-marketplace:ResolveCustomer"  
            ],  
            "Resource": "*",  
            "Effect": "Allow"  
        }  
    ]  
}
```

Política de relacionamento confiável da função de medição

```
{  
    "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 a esse código de produto do AWS Marketplace. Você pode habilitar a associação configurando o campo `productCode` do `UsagePlan` do API Gateway para seu código do produto do AWS Marketplace usando o console do API Gateway,

a API REST do API Gateway, a AWS CLI do 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 assina seu produto pelo AWS Marketplace, o AWS Marketplace encaminha uma solicitação `POST` ao URL de assinaturas de SaaS que você registrou ao listar seu produto no AWS Marketplace. A solicitação `POST` acompanha um parâmetro de cabeçalho `x-amzn-marketplace-token` contendo informações do comprador. Siga as instruções em [Configurar seu produto de SaaS para aceitar novos clientes](#) para lidar com esse redirecionamento em seu aplicativo do portal do desenvolvedor.

Ao responder à solicitação de assinatura de um cliente, o AWS Marketplace envia uma notificação `subscribe-success` para um tópico do Amazon SNS ao qual você pode se inscrever. (Consulte [Configurar o produto de SaaS para aceitar novos clientes](#)). Para aceitar a solicitação de assinatura do cliente, manipule 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 de API e, em seguida, adicionando a chave de 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 determinado cliente, use a API do API Gateway para buscar ou criar uma chave de API para o cliente e adicione essa chave de API a esse 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 `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 customerId específico do AWS Marketplace (**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 uma API de exemplo keyId retornada 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 seguinte exemplo de código 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"
        }
    ]
}
```

Configurar um nome de domínio personalizado para uma API no API Gateway

Note

Agora é possível escolher uma versão mínima do TLS que seja compatível com a API. Para APIs REST, é possível escolher TLS 1.2 ou TLS 1.0. Para APIs do WebSocket, é possível escolher somente TLS 1.2.

Após a implantação da sua API REST ou WebSocket, 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 especificado por você ao criar a API e **stage** é especificado por você ao implantar a API.

Note

Um domínio personalizado pode ser associado a uma API REST ou a uma API WebSocket, mas não a ambas.

Os nomes de domínio personalizados não são compatíveis com [APIs privadas \(p. 200\)](#).

A parte do nome de host do URL (ou seja, **api-id**.execute-api.**region**.amazonaws.com) refere-se a um endpoint de API, que pode ser otimizado para bordas ou regional. O endpoint de API padrão pode ser difícil de chamar novamente e pode não ser amigável. Para fornecer um URL mais simples e intuitivo aos usuários da API, é possível configurar um nome de domínio personalizado (por exemplo, **api.example.com**) como o nome do host da API e escolher um caminho de base (por exemplo, **myservice**) para mapear o URL alternativo para essa API. A URL de base de API mais amigável agora se torna:

```
https://api.example.com/myservice
```

Se você não definir nenhum mapeamento de base sob um nome de domínio personalizado, o URL de base da API resultante será a mesma que o domínio personalizado (por exemplo, <https://api.example.com>). Nesse caso, o nome do domínio personalizado não pode oferecer suporte a mais de uma API.

Quando você implanta uma API otimizada para fronteiras, o API Gateway configura uma distribuição do Amazon CloudFront e um registro DNS para mapear o nome de domínio da API para o nome de domínio de distribuição do CloudFront. Solicitações para a API são roteadas para o API Gateway por meio da distribuição mapeada do CloudFront.

Quando você cria um nome de domínio personalizado para uma API otimizada para fronteiras, o API Gateway configura uma distribuição do CloudFront. No entanto, você deve configurar um registro DNS para mapear o nome de domínio personalizado para o nome de domínio de distribuição do CloudFront para as solicitações de API que são direcionadas para o nome de domínio personalizado, as quais são roteadas para o API Gateway por meio da distribuição mapeada do CloudFront. Você também deve fornecer um certificado para o nome de domínio personalizado.

Quando você cria um nome de domínio personalizado para uma API regional, o API Gateway cria um nome de domínio regional para a API. Você deve configurar um registro DNS para mapear o nome de domínio personalizado para o nome de domínio regional para as solicitações de API que são direcionadas para o nome de domínio personalizado, as quais são roteadas para o API Gateway por meio do endpoint de API regional mapeado. Você também deve fornecer um certificado para o nome de domínio personalizado.

Note

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 CloudWatch Logs, é necessário usar esse ID da conta do API Gateway. Para obter mais informações, consulte [Registrar a criação do nome de domínio personalizado no CloudTrail \(p. 689\)](#).

Para configurar um nome de domínio personalizado otimizado para fronteiras ou atualizar seu certificado, você deve ter uma permissão para atualizar as distribuições do CloudFront. Isso pode ser feito anexando a seguinte instrução de política do IAM a um usuário, um grupo ou uma função do IAM na conta:

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

```
    "Statement": [
        {
            "Sid": "AllowCloudFrontUpdateDistribution",
            "Effect": "Allow",
            "Action": [
                "cloudfront:updateDistribution"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

O API Gateway oferece suporte a nomes de domínio personalizados otimizados para fronteiras, otimizando a Indicação de nome de servidor (SNI) na distribuição do CloudFront. Para obter mais informações sobre o uso de nomes de domínio personalizados em uma distribuição do CloudFront, incluindo o formato de certificado necessário e o tamanho máximo de um comprimento de chave de certificado, consulte [Como usar nomes de domínio alternativos e HTTPS](#) no documento Guia do desenvolvedor do Amazon CloudFront.

Para configurar um nome de domínio personalizado como o nome de host da API, você, como proprietário da API, deve fornecer um certificado SSL/TLS para o nome de domínio personalizado.

Para fornecer um certificado para um nome de domínio personalizado otimizado para bordas, é possível solicitar que o [AWS Certificate Manager](#) (ACM) gere um novo certificado no ACM ou importe para o ACM um outro emitido por uma autoridade de certificação de terceiros na região `us-east-1` (Norte da Virgínia).

Para fornecer um certificado para um nome de domínio personalizado regional em uma região onde o ACM é compatível, é necessário solicitar um certificado do ACM. Para fornecer um certificado para um nome de domínio personalizado regional em uma região onde não há suporte para o ACM, é 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. Para usar um certificado gerenciado pela AWS para um nome de domínio, basta fazer referência ao seu ARN.

O ACM simplifica a configuração e o uso de um nome de domínio personalizado para uma API: basta criar ou importar para o ACM um certificado para o nome de domínio especificado, configurar esse nome de domínio no API Gateway com o ARN do certificado fornecido pelo ACM e mapear um caminho base sob o nome de domínio personalizado para um estágio implantado da API. Com certificados emitidos pelo ACM, você não precisa se preocupar em expor detalhes de certificados confidenciais, como a chave privada.

É necessário ter um nome de domínio da Internet registrado para configurar nomes de domínio personalizados para as APIs. Se necessário, é possível registrar um domínio da Internet usando o [Amazon Route 53](#) ou um registrador de domínios de terceiros da sua escolha. O nome de domínio personalizado de uma API pode ser o nome de um subdomínio ou do domínio raiz (também conhecido como apex de zona) de um domínio da Internet registrado.

Depois que um nome de domínio personalizado é criado no API Gateway, você deve criar ou atualizar o registro de recurso do provedor do serviço de nome de domínio (DNS) para mapear o nome de domínio personalizado otimizado para fronteiras para o nome de domínio da distribuição do CloudFront ou para mapear o nome de domínio personalizado regional para o endpoint de API regional. Sem esse mapeamento, as solicitações de API que forem direcionadas para o nome de domínio personalizado não poderão acessar o API Gateway.

Note

Um nome de domínio personalizado otimizado para bordas é criado em uma região específica e pertence a determinada conta da AWS. A movimentação desse nome de domínio personalizado

entre regiões ou contas da AWS envolve a exclusão da distribuição existente do CloudFront e a criação de outra. O processo pode levar aproximadamente 30 minutos antes que o novo nome de domínio personalizado torna-se disponível. Para obter mais informações, consulte [Atualizando distribuições do CloudFront](#).

As páginas a seguir descrevem como usar o ACM para criar um certificado SSL/TLS para um nome de domínio personalizado, configurar o nome de domínio personalizado para uma API, associar uma API específica a um caminho base sob o nome de domínio personalizado e para renovar (ou seja, revezar) um certificado prestes a expirar que foi importado para o ACM para o nome de domínio personalizado.

Nomes de domínio personalizados curinga

O API Gateway também oferece suporte a nomes de domínio personalizados curinga. É possível especificar 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. O assunto do certificado deve ser o nome de domínio personalizado curinga.

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 compartilharem um proprietário, será possível entrar em contato com o [AWS Support Center](#) para solicitar uma exceção.

Tópicos

- [Preparar certificados para uso no AWS Certificate Manager \(p. 679\)](#)
- [Escolher uma versão mínima do TLS para um domínio personalizado no API Gateway \(p. 682\)](#)
- [Como criar um nome de domínio personalizado otimizado para fronteiras \(p. 685\)](#)
- [Configurar um nome de domínio personalizado para uma API WebSocket ou uma API REST regional no API Gateway \(p. 692\)](#)
- [Migrar um nome de domínio personalizado para outro endpoint de API \(p. 697\)](#)

Preparar certificados para uso 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. As etapas a seguir descrevem como fazer isso. Para obter mais informações, consulte o [Guia do usuário do AWS Certificate Manager](#).

Note

Para usar um certificado do ACM com um nome de domínio personalizado otimizado para fronteiras do API Gateway, você deve solicitar ou importar o certificado na região Leste dos

EUA (Norte da Virgínia) (`us-east-1`). Para um nome de domínio personalizado regional do API Gateway, você deve solicitar ou importar o certificado na mesma região da sua API.

Para obter um certificado para um determinado nome de domínio emitido pelo ou importados para o ACM

1. Registre seu domínio da Internet, por exemplo, `myDomain.com`. Você pode usar um [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.
2. Para criar ou importar um certificado SSL/TLS para o ACM para um nome de domínio, siga um destes procedimentos:

Para solicitar um certificado fornecido pelo ACM para um nome de domínio

1. Faça login no [console do AWS Certificate Manager](#).
2. Selecione Request a certificate.
3. Digite um nome de domínio personalizado para a sua API; por exemplo, `api.example.com`, em Domain name.
4. Opcionalmente, escolha Add another name to this certificate.
5. Escolha Review and request.
6. Escolha Confirm and request.
7. Para uma solicitação válida, um proprietário registrado do domínio da Internet deve concordar com a solicitação antes, e o ACM emite o certificado.

Para importar para o ACM um certificado para um nome de domínio

1. Obtenha um certificado SSL/TLS codificado em PEM para seu nome de domínio personalizado de uma autoridade de certificação (CA). Para obter uma lista parcial desses CAs, consulte a [Lista de CAs incluídas no Mozilla](#)
 - 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
```

Note

O Amazon API Gateway utiliza o Amazon CloudFront para oferecer suporte a certificados para nomes de domínio personalizados. Como tal, os requisitos e 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](#).

- 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. Para Certificate body, digite ou cole o corpo do certificado de servidor com formato PEM da sua autoridade de certificação. Veja a seguir um exemplo abreviado desse tipo de certificado.

```
-----BEGIN CERTIFICATE-----  
EXAMPLECA+KgAwIBAgIQJ1XxJ8P1++gOfQtj0IBoqDANBgkqhkiG9w0BAQUFADBB  
...  
az8Cg1aicxLBQ7EaWIhhgEXAMPLE  
-----END CERTIFICATE-----
```

- d. Para o Certificate private key, digite ou cole seu PEM chave privada do certificado formatado. Veja a seguir um exemplo abreviado desse tipo de chave.

```
-----BEGIN RSA PRIVATE KEY-----  
EXAMPLEBAAKCAQEA2Qb3LDHD7StY7Wj6U2/opV6Xu37qUCCkeDWhwpZMYJ9/nETO  
...  
1qGvJ3u04vdnzaYN5WoyN5LFckrlA71+CszD1CGSqbVDWEXAMPLE  
-----END RSA PRIVATE KEY-----
```

- e. Para Certificate chain, digite ou cole os certificados intermediários com 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/ifBLIK3se2e4/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 Revisar e importar.

- 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.

Escolher uma versão mínima do TLS para um domínio personalizado no API Gateway

Para obter maior segurança, é possível escolher uma versão mínima do protocolo Transport Layer Security (TLS) para ser aplicada ao domínio personalizado do Amazon API Gateway configurando uma política de segurança no console do API Gateway, na AWS CLI ou nos SDKs da AWS.

Uma política de segurança é uma combinação predefinida da versão mínima do TLS e do pacote de criptografia oferecida pelo Amazon API Gateway. É possível escolher uma política de segurança do TLS versão 1.2 ou do TLS versão 1.0. 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.

Nas configurações do domínio personalizado, uma política de segurança determina duas configurações:

- A versão mínima do TLS que o API Gateway usa para se comunicar com clientes da API
- A criptografia que o API Gateway usa para criptografar o conteúdo que ele retorna aos clientes da API

Tópicos

- [Como especificar uma versão mínima do protocolo TLS para domínios personalizados no API Gateway \(p. 682\)](#)
- [Políticas de segurança, versões do protocolo TLS e criptografias compatíveis com endpoints de API otimizados para fronteiras no API Gateway \(p. 683\)](#)
- [Criptografias e protocolos SSL/TLS compatíveis para endpoints de API do WebSocket, privados e regionais no API Gateway \(p. 683\)](#)
- [Nomes das criptografias OpenSSL e RFC \(p. 684\)](#)

Como especificar uma versão mínima do protocolo TLS para domínios personalizados no API Gateway

Ao criar um domínio personalizado, especifique a política de segurança para ele. Para obter mais informações sobre políticas de segurança, consulte as tabelas nas seções a seguir.

As seções a seguir descrevem como criar um nome de domínio personalizado, incluindo a especificação da versão mínima do TLS no console do API Gateway e na CLI:

- [the section called “Como criar um nome de domínio personalizado otimizado para fronteiras” \(p. 685\)](#)
- [the section called “Configurar um nome de domínio personalizado regional para a API WebSocket ou REST” \(p. 692\)](#)

É possível alterar a política de segurança atualizando as configurações do nome de domínio. Para alterar a versão mínima do TLS, use um dos comandos a seguir, especificando a nova versão do TLS (`TLS_1_0` ou `TLS_1_2`) no parâmetro `securityPolicy`. Aguarde até 60 minutos para que a atualização seja concluída.

- [`domainname:update`](#)

- [update-domain-name](#)

Políticas de segurança, versões do protocolo TLS e criptografias compatíveis com endpoints de API otimizados para fronteiras no API Gateway

A tabela a seguir lista as criptografias e os protocolos que o API Gateway pode usar para cada política de segurança de APIs otimizadas para fronteiras.

	Política de segurança	
TLSv1.2	◆	◆
TLSv1.1	◆	
TLSv1	◆	
SSLv3		
Ciphers Supported		
ECDHE-RSA-AES128-GCM-SHA256	◆	◆
ECDHE-RSA-AES128-SHA256	◆	◆
ECDHE-RSA-AES128-SHA	◆	
ECDHE-RSA-AES256-GCM-SHA384	◆	◆
ECDHE-RSA-AES256-SHA384	◆	◆
ECDHE-RSA-AES256-SHA	◆	
AES128-GCM-SHA256	◆	◆
AES256-GCM-SHA384	◆	◆
AES128-SHA256	◆	◆
AES256-SHA	◆	
AES128-SHA	◆	
DES-CBC3-SHA	◆	
RC4-MD5		

Criptografias e protocolos SSL/TLS compatíveis para endpoints de API do WebSocket, privados e regionais no API Gateway

A tabela a seguir descreve as políticas de segurança que podem ser especificadas para endpoints de API do WebSocket, privados e regionais.

Note

Para APIs do WebSocket e privados, somente `TLS-1-2` pode ser especificado.

Política de segurança	TLS-1-0	TLS-1-2
Protocolos TLS		
Protocol-TLSv1	◆	
Protocol-TLSv1.1	◆	
Protocol-TLSv1.2	◆	◆
Cifras TLS		
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-RSA-AES256-SHA	◆	
ECDHE-ECDSA-AES256-SHA	◆	
AES128-GCM-SHA256	◆	◆
AES128-SHA256	◆	◆
AES128-SHA	◆	
AES256-GCM-SHA384	◆	◆
AES256-SHA256	◆	◆
AES256-SHA	◆	
DES-CBC3-SHA	◆	

Nomes das criptografias OpenSSL e RFC

OpenSSL e IETF RFC 5246, o protocolo Transport Layer Security (TLS) versão 1.2, use nomes diferentes para as mesmas criptografias. A tabela a seguir mapeia o nome do OpenSSL para o nome do RFC para cada criptograma.

Nome da criptografia OpenSSL	Nome da criptografia RFC
ECDHE-RSA-AES128-GCM-SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
ECDHE-RSA-AES128-SHA256	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
ECDHE-RSA-AES128-SHA	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
ECDHE-RSA-AES256-GCM-SHA384	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
ECDHE-RSA-AES256-SHA384	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
ECDHE-RSA-AES256-SHA	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
AES128-GCM-SHA256	TLS_RSA_WITH_AES_128_GCM_SHA256
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
RC4-MD5	TLS_RSA_WITH_RC4_128_MD5

Como criar um nome de domínio personalizado otimizado para fronteiras

Tópicos

- [Configurar um nome de domínio personalizado otimizado para fronteiras para uma API do API Gateway \(p. 686\)](#)
- [Registrar a criação do nome de domínio personalizado no CloudTrail \(p. 689\)](#)
- [Configurar o mapeamento de caminho base de uma API com um nome de domínio personalizado como seu nome de host \(p. 689\)](#)
- [Revezar um certificado importado para o ACM \(p. 690\)](#)
- [Chamar a API com nomes de domínio personalizado \(p. 691\)](#)

Configurar um nome de domínio personalizado otimizado para fronteiras para uma API do API Gateway

O procedimento a seguir descreve como configurar um nome de domínio personalizado para uma API usando o console do API Gateway.

Para configurar um nome de domínio personalizado usando o console do API Gateway

1. Faça login no console do API Gateway em <https://console.aws.amazon.com/apigateway>.
2. Escolha Custom Domain Names no painel de navegação principal.
3. Escolha Create Custom Domain Name na sequência.
4. Em New Custom Domain Name (Novo nome de domínio personalizado):
 - a. Para o protocolo da API, selecione HTTP.
 - b. Em Domain Name (Nome de domínio), digite um nome de domínio personalizado, por exemplo, `my-api.example.com`, em Domain Name (Nome de domínio).
 - c. Em Security Policy (Política de segurança), escolha a versão mínima desejada do Transport Layer Security (TLS).
 - d. Em Endpoint Configuration (Configuração do endpoint), selecione Edge optimized (Otimizado para fronteiras).
 - e. Escolha um certificado na lista ACM Certificate (Certificados do ACM).

Note

Para usar um certificado do ACM com um nome de domínio personalizado otimizado para fronteiras do API Gateway, você deve solicitar ou importar o certificado na região `us-east-1` (Norte da Virgínia).

- f. Escolha Add mapping em Base Path Mappings para definir um caminho base (Path) para um API implementado em determinado estágio (selecionado na lista suspensa Destination.) Você também pode definir o mapeamento do caminho base após a criação do nome de domínio personalizado. Para obter mais informações, consulte [Configurar o mapeamento de caminho base de uma API com um nome de domínio personalizado como seu nome de host \(p. 689\)](#).
 - g. Escolha Save (Salvar).
5. Após a criação do nome de domínio personalizado, o console exibe o nome de domínio da distribuição do CloudFront associado, no formato `distribution-id.cloudfront.net`, junto com o ARN do certificado. Observe o nome de domínio da distribuição do CloudFront mostrado na saída. Você precisará disso na próxima etapa para definir o valor CNAME do domínio personalizado ou o destino do alias do registro A no seu DNS.

Note

O nome de domínio personalizado recém-criado leva cerca de 40 minutos para ficar pronto. Enquanto isso, você pode configurar o alias de registro DNS para mapear o nome de domínio personalizado para o nome de domínio de distribuição do CloudFront associado e configurar o mapeamento do caminho base para o nome de domínio personalizado enquanto este último está sendo inicializado.

6. Nesta etapa, usamos o Amazon Route 53 como exemplo de provedor de DNS para mostrar como configurar um alias de registro A para o seu domínio da Internet, com o objetivo de mapear o nome de domínio personalizado para o nome da distribuição do CloudFront associado. As instruções podem ser adaptadas para outros provedores de DNS.
 - a. Faça login no console do Route 53.
 - b. Crie um conjunto de registros A-IPv4 address para seu domínio personalizado (por exemplo, `api.example.com`). Um registro A mapeia um nome de domínio personalizado para um endereço IP4.

- c. Escolha Yes para Alias, digite o nome de domínio CloudFront (por exemplo, d3boq9ikothtgw.cloudfront.net) em Alias Target e escolha Create. Aqui, o alias de registro A mapeia seu nome de domínio personalizado para o nome de domínio do CloudFront especificado que ele próprio mapeou para um endereço IP4.

Create Record Set

Name: api.haymuto.com.

Type: A – IPv4 address

Alias: Yes No

Alias Target: d3boq9ikothtgw.cloudfront.net

Alias Hosted Zone ID: Z2FDTNDATAQYW2

You can also type the domain name for the resource. Examples:
- CloudFront distribution domain name: d111111abcdef8.cloudfront.net
- Elastic Beanstalk environment CNAME: example.elasticbeanstalk.com
- ELB load balancer DNS name: example-1.us-east-1.elb.amazonaws.com
- S3 website endpoint: s3-website.us-east-2.amazonaws.com
- Resource record set in this hosted zone: www.example.com

[Learn More](#)

Routing Policy: Simple

Route 53 responds to queries based only on the values in this record. [Learn More](#)

Evaluate Target Health: Yes No

Create

Tip

A Alias Hosted Zone ID identifica a zona hospedada do Alias Target especificado. O console do Route 53 preenche o valor automaticamente quando você insere um nome de domínio válido para Alias Target. Para criar um alias de registro A sem usar o console do Route 53, como quando você usa a AWS CLI, é necessário especificar o ID da zona hospedada necessário. Para qualquer nome de domínio de distribuição do CloudFront, o valor de ID da zona hospedada é sempre Z2FDTNDATAQYW2, conforme documentado em [Regiões e endpoints da AWS para o CloudFront](#).

Para a maioria dos provedores de DNS, um nome de domínio personalizado é adicionado à zona hospedada como um conjunto de registros de recursos CNAME. O nome do registro CNAME especifica o nome de domínio personalizado que você digitou anteriormente em Domain Name (por exemplo, api.example.com). O valor do registro CNAME especifica o nome do domínio para a distribuição do CloudFront. No entanto, o uso de um registro CNAME não funcionará se o seu domínio personalizado for um ápice de zona (ou seja, example.com em vez de api.example.com). Um ápice de zona também é conhecido como o domínio raiz da sua organização. Para um ápice de zona, você precisa usar um alias de registro A, desde que ele tenha suporte pelo seu provedor de DNS.

Com o Route 53 você pode criar um alias de registro A para seu nome de domínio personalizado e especificar o nome de domínio da distribuição do CloudFront como o destino do alias, conforme mostrado acima. Isso significa que o Route 53 pode rotear seu nome de domínio personalizado, mesmo que seja um ápice de zona. Para obter mais informações, consulte [Escolhendo entre conjuntos de registros de recursos de alias e não alias](#) no Guia do desenvolvedor do Amazon Route 53.

O uso de alias de registros A também elimina a exposição do nome de domínio da distribuição do CloudFront subjacente, pois o mapeamento de nome de domínio ocorre exclusivamente no Route 53. Por esses motivos, recomendamos que você use o alias de registro A do Route 53 sempre que possível.

Além de usar o console do API Gateway, você pode usar a API REST do API Gateway, a CLI da AWS ou um dos SDKs da AWS para configurar o nome de domínio personalizado das suas APIs. Como ilustração, o procedimento a seguir descreve as etapas para fazer isso usando as chamadas da API REST.

Para configurar um nome de domínio personalizado usando a API REST do API Gateway

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, bem como o nome da distribuição do CloudFront associada em sua carga.

2. Observe o nome de domínio da distribuição do CloudFront mostrado na saída. Você precisará disso na próxima etapa para definir o valor CNAME do domínio personalizado ou o destino do alias do registro A no seu DNS.
3. Siga a Etapa 6 do procedimento anterior para configurar um alias de registro A de modo a mapear o nome de domínio personalizado para seu nome de distribuição do CloudFront.

Para exemplos de código dessa chamada de API REST, consulte [domainname:create](#).

Registrar a criação do nome de domínio personalizado no CloudTrail

Quando o CloudTrail está habilitado para registrar 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. 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-north-1	394634713161
ap-northeast-1	969236854626
ap-northeast-2	020402002396
ap-southeast-1	195145609632
ap-southeast-2	798376113853
ap-south-1	507069717855
ap-east-1	174803364771
sa-east-1	287228555773
me-south-1	855739686837

Configurar o mapeamento de caminho base de uma API com um nome de domínio personalizado como seu nome de host

Você pode usar um único nome de domínio personalizado como o nome do host de várias APIs. Para fazer isso, configure os mapeamentos de caminho base no nome de domínio personalizado. Com os mapeamentos de caminho base, uma API no domínio personalizado é acessível por meio da combinação de nome de domínio personalizado e do caminho base associado.

Por exemplo, se você tiver criado uma API chamada PetStore e outra API chamada PetShop e configurar um nome de domínio personalizado de `api.example.com` no API Gateway, poderá definir

a URL da API PetStore como `https://api.example.com` ou `https://api.example.com/myPetStore`. A API PetStore está associada ao caminho base de uma string vazia ou a myPetStore no nome de domínio personalizado de `api.example.com`. Da mesma forma, você pode atribuir um caminho base de `yourPetShop` para a API PetShop. A URL de `https://api.example.com/yourPetShop` é então a URL raiz da API PetShop.

Antes de definir o caminho base para uma API, conclua as etapas em [Configurar um nome de domínio personalizado otimizado para fronteiras para uma API do API Gateway \(p. 686\)](#).

Para definir o caminho base para mapeamentos de API usando o console do API Gateway

1. Escolha um nome de domínio personalizado na lista de Custom Domain Names disponíveis na sua conta.
2. Escolha Show Base Path Mappings ou Edit.
3. Escolha Add mapping.
4. (Opcional) Digite um nome de caminho base para Path, escolha uma API de Destination e escolha um estágio.

Note

A lista Destination mostra as APIs implantadas na sua conta.

5. Escolha Save para concluir a configuração do mapeamento de caminho base para a API.

Note

Para excluir um mapeamento depois de criá-lo, ao lado do mapeamento que você deseja excluir, escolha o ícone de lixeira.

Além disso, você pode chamar a API REST do API Gateway, a CLI da AWS ou um dos SDKs da AWS para configurar o mapeamento de caminho base de uma API com um nome de domínio personalizado como seu nome de host. Como ilustração, o procedimento a seguir descreve as etapas para fazer isso usando as chamadas da API REST.

Para configurar o mapeamento do caminho base de uma API usando a API REST do API Gateway

- Chame `basepathmapping:create` em um nome de domínio personalizado, especificando `basePath`, `restApiId` e uma propriedade `stage` de implantação na carga 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](#).

Revezar um certificado importado para o ACM

O ACM lida automaticamente com a renovação dos certificados que ele emite. Não é necessário revezar 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 revezar 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 revezar esse certificado existente para o novo certificado

emitido pelo ACM. Depois disso, é possível deixar o ACM e o CloudFront encarregados de lidar com o revezamento de certificados para você automaticamente. Para criar ou importar um novo certificado do ACM, siga as etapas para [solicitar ou importar um novo certificado do ACM \(p. 679\)](#) para o nome de domínio especificado.

Para revezar um certificado para um nome de domínio, você pode usar o console do API Gateway, a API REST do API Gateway, a CLI da AWS ou um dos SDKs da AWS.

Para revezar um certificado prestes a expirar importado para o ACM usando o console do API Gateway

1. Solicite ou importe um certificado no ACM.
2. Retorne ao console do API Gateway.
3. Escolha Custom Domain Names no painel de navegação principal do console API Gateway.
4. Selecione o nome de domínio personalizado desejado no painel Custom Domain Names.
5. Selecione Edit.
6. Escolha o certificado desejado na lista suspensa ACM Certificate.
7. Escolha Save para começar a revezar o certificado para o nome de domínio personalizado.

Note

Demora cerca de 40 minutos para terminar o processo. Após o revezamento, você pode escolher o ícone de seta bidirecional ao lado de ACM Certificate para reverter para o certificado original.

Para ilustrar como revezar programaticamente um certificado importado para um nome de domínio personalizado, descrevemos as etapas usando a API REST do API Gateway.

Revezar um certificado importado usando a API REST do API Gateway

- Chame a ação [domainname:update](#), especificando o ARN do novo certificado do ACM para o nome de domínio especificado.

Chamar a API com nomes de domínio personalizado

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`).

URLs raiz de APIs com nomes de domínio padrão e personalizados

ID de API	Estágio	URL padrão	Caminho base	URL personalizado
udxjef	pro	<code>https://udxjef.execute-api.us-east-1.amazonaws.com/pro</code>	/petstore	<code>https://api.example.com/petstore</code>
udxjef	tst	<code>https://udxjef.execute-api.us-</code>	/petdepot	<code>https://api.example.com/petdepot</code>

ID de API	Estágio	URL padrão	Caminho base	URL personalizado
		east-1.amazonaws.com/tst		
qf3duz	dev	https://qf3duz.execute-api.us-east-1.amazonaws.com/dev	/bookstore	https://api.example.com/bookstore
qf3duz	tst	https://qf3duz.execute-api.us-east-1.amazonaws.com/tst	/bookstand	https://api.example.com/bookstand

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 a SNI na distribuição do CloudFront. Para obter informações sobre como o CloudFront usa nomes de domínio personalizados, consulte [Amazon CloudFront Custom SSL](#).

Configurar um nome de domínio personalizado para uma API WebSocket ou uma API REST regional no API Gateway

Assim como ocorre com um endpoint de API otimizado para fronteiras, você pode criar um nome de domínio personalizado para um endpoint de API regional. Para oferecer suporte a um nome de domínio personalizado regional, você deve fornecer um certificado. Se um certificado do AWS Certificate Manager (ACM) for usado, esse certificado deverá ser específico da região. Se o ACM estiver disponível na região, você deverá fornecer um certificado do ACM específico dessa região. Se o ACM não tiver suporte nessa região, você deverá fazer upload de um certificado para o API Gateway nessa região ao criar o nome de domínio personalizado regional. Para obter mais informações sobre a criação ou upload de um certificado de nome de domínio personalizado, consulte [Preparar certificados para uso no AWS Certificate Manager \(p. 679\)](#).

Important

Para um nome de domínio personalizado regional do API Gateway, você deve solicitar ou importar o certificado na mesma região da sua API.

Quando você cria um nome de domínio regional personalizado (ou migra um) com um certificado do ACM, o API Gateway cria uma função vinculada a um serviço na sua conta, se a função ainda não existir. A função vinculada a um serviço é necessária para anexar seu certificado do ACM ao seu endpoint regional. A função é denominada `AWSServiceRoleForAPIGateway` e terá a política gerenciada `APIGatewayServiceRolePolicy` anexada. Para obter mais informações sobre como usar a função vinculada a um serviço, consulte [Como usar funções vinculadas a serviços](#).

Important

Você deve criar um registro DNS para apontar o nome de domínio personalizado para o nome de domínio regional. Isso permite que o tráfego que está vinculado ao nome de domínio personalizado seja direcionado para o nome de host regional da API. O registro DNS pode ser do tipo CNAME ou A.

Tópicos

- [Configurar um nome de domínio regional personalizado com certificado do ACM usando o console do API Gateway \(p. 693\)](#)
- [Configurar um nome de domínio personalizado regional com certificado do ACM usando a AWS CLI \(p. 693\)](#)
- [Configurar o certificado de um nome de domínio personalizado regional sem o ACM usando a AWS CLI \(p. 695\)](#)

Configurar um nome de domínio regional personalizado com certificado do ACM usando o console do API Gateway

Para usar o console do API Gateway para configurar um nome de domínio regional, use este procedimento.

Para configurar um nome de domínio regional usando o console do API Gateway

1. Faça login no console do API Gateway e selecione Custom Domain Names (Nomes de domínio personalizados) no painel de navegação principal.
2. Escolha +Create New Custom Domain Name (+Criar novo nome de domínio personalizado) acima da tabela Custom Domain Names (Nomes de domínio personalizados).
3. Em New Custom Domain Name (Novo nome de domínio personalizado):
 - a. Escolha o protocolo de API: HTTP ou WebSocket.
 - b. Em Domain Name (Nome de domínio), digite um nome de domínio personalizado, por exemplo, `my-api.example.com`, em Domain Name (Nome de domínio).
4. Em Security Policy (Política de segurança), escolha a versão mínima desejada do Transport Layer Security (TLS).
5. Em Endpoint Configuration (Configuração do endpoint), selecione Regional.
6. Selecione um certificado na lista suspensa ACM Certificate (Certificado do ACM). O certificado deve estar na mesma região em que a API está implantada.
7. Se você tiver criado e implantado uma API para usar esse nome de domínio personalizado, escolha Add mapping (Adicionar mapeamento), digite um caminho de base sob o nome de domínio personalizado em Path (Caminho), escolha uma API na lista suspensa de APIs em Destination (Destino) e escolha uma etapa na lista suspensa Stage (Estágio). Para adicionar outro mapeamento de caminho de base, repita a etapa. Você também pode definir o mapeamento do caminho base após a criação do nome de domínio personalizado. Para obter mais informações, consulte [Configurar o mapeamento de caminho base de uma API com um nome de domínio personalizado como seu nome de host \(p. 689\)](#).
8. Escolha Save (Salvar).
9. Siga a documentação do Route 53 sobre [como configurar o Route 53 para rotear tráfego para o API Gateway](#).

Configurar um nome de domínio personalizado regional com certificado do ACM usando a AWS CLI

Para usar a AWS CLI para configurar um nome de domínio personalizado para uma API regional, use este procedimento.

1. Chame `create-domain-name`, especificando um nome de domínio personalizado do tipo REGIONAL e o ARN (Nome de recurso da Amazon) de um certificado regional.

```
aws apigateway create-domain-name \
--domain-name 'regional.example.com' \
--endpoint-configuration types=REGIONAL \
--regional-certificate-arn 'arn:aws:acm:us-west-2:123456789012:certificate/
c19332f0-3be6-457f-a244-e03a423084e6'
```

Observe que o certificado especificado é da região us-west-2 e para este exemplo, vamos considerar que a API subjacente está na mesma região.

Se houver êxito, a chamada retornará um resultado semelhante ao seguinte:

```
{  
    "certificateUploadDate": "2017-10-13T23:02:54Z",  
    "domainName": "regional.example.com",  
    "endpointConfiguration": {  
        "types": "REGIONAL"  
    },  
    "regionalCertificateArn": "arn:aws:acm:us-west-2:123456789012:certificate/
c19332f0-3be6-457f-a244-e03a423084e6",  
    "regionalDomainName": "d-numh1z56v6.execute-api.us-west-2.amazonaws.com"  
}
```

O valor da propriedade `regionalDomainName` 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 que está vinculado ao nome de domínio personalizado seja direcionado para o nome de host dessa API regional.

2. Crie um registro DNS pra associar o nome de domínio personalizado e o nome de domínio regional. Isso permite que solicitações que estão vinculadas ao nome de domínio personalizado sejam direcionadas para o nome de host regional da API.
3. Adicione um mapeamento de caminho base para expor a API especificada (por exemplo, `0qzs2sy7bh`) em uma etapa de implantação (por exemplo, `test`) com o nome de domínio personalizado especificado (por exemplo, `regional.example.com`).

```
aws apigateway create-base-path-mapping \
--domain-name 'regional.example.com' \
--base-path 'RegionalApiTest' \
--rest-api-id 0qzs2sy7bh \
--stage 'test'
```

Como resultado, a URL base usando o nome de domínio personalizado para a API que é implantada na etapa se torna `https://regional.example.com/RegionalApiTest`.

4. Configure seus registros DNS para mapear o nome de domínio regional personalizado ao seu nome de host do ID de zona hospedada. Primeiro, crie o arquivo JSON que contém a definição para a configuração de um registro DNS para o nome de domínio regional. O exemplo a seguir mostra como criar um registro A DNS regional para mapear um nome de domínio personalizado (`regional.example.com`) ao seu 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. Você também pode obter os IDs de zona hospedada regional do Route 53 em [Regiões e endpoints do API Gateway](#).

```
{  
    "Changes": [  
        {  
            "Action": "CREATE",  
            "Resource": {  
                "Type": "AWS::Route53::RecordSet",  
                "Properties": {  
                    "Name": "d-numh1z56v6.execute-api.us-west-2.amazonaws.com",  
                    "Type": "A",  
                    "TTL": 300,  
                    "AliasTarget": {  
                        "HostedZoneId": "Z2FDTNDATAQYW2",  
                        "DNSName": "d-numh1z56v6.execute-api.us-west-2.amazonaws.com",  
                        "EvaluateTargetHealth": false  
                    }  
                }  
            }  
        }  
    ]  
}
```

```
"ResourceRecordSet": [
    "Name": "regional.example.com",
    "Type": "A",
    "AliasTarget": {
        "DNSName": "d-numh1z56v6.execute-api.us-west-2.amazonaws.com",
        "HostedZoneId": "Z2OJLYMU09EFXC",
        "EvaluateTargetHealth": false
    }
]
}
```

5. Execute o seguinte comando da CLI:

```
aws route53 change-resource-record-sets \
--hosted-zone-id {your-hosted-zone-id} \
--change-batch file://path/to/your/setup-dns-record.json
```

onde `{your-hosted-zone-id}` é o ID de zona hospedada do Route 53 do conjunto de registros DNS em sua conta. O valor do parâmetro `change-batch` aponta para um arquivo JSON (`setup-dns-record.json`) em uma pasta (`path/to/your`).

Configurar o certificado de um nome de domínio personalizado regional sem o ACM usando a AWS CLI

Para usar a AWS CLI a fim de configurar um nome de domínio personalizado para uma API regional em regiões onde o ACM não está disponível, será necessário enviar o certificado diretamente ao API Gateway usando o procedimento a seguir.

Para fazer upload de um certificado de servidor para o API Gateway, é necessário fornecer o certificado e sua chave privada correspondente. quando o certificado não for autoassinado, você também deverá fornecer uma cadeia de certificado. (Você não precisa de uma cadeia de certificado ao fazer upload de um certificado autoassinado.) Antes de fazer upload de um certificado, verifique se você tem todos estes itens e que atendem aos seguintes critérios:

- O certificado deve ser válido no momento do upload. Você não pode fazer upload de um certificado antes de seu período de validade começar (a data `NotBefore` do certificado) ou após sua expiração (a data `NotAfter` do certificado).
- A chave privada deve ser não criptografada. Você não pode fazer upload de uma chave privada que seja protegida por uma senha ou código de acesso. Para ajudar a descriptografar uma chave privada criptografada, consulte [Solução de problemas](#).
- O certificado, a chave privada e a cadeia de certificação devem ser codificados em PEM. Para ajudar a converter esses itens para o formato PEM, consulte [Solução de problemas](#).

1. Crie os seguintes arquivos PEM: Se você usar nomes de arquivos diferentes, será necessário alterar os nomes de arquivos no comando `create-domain-name` para que sejam correspondentes.
 - Armazene o certificado codificado por PEM em um arquivo chamado `Certificate.pem`.
 - Armazene a cadeia de certificados codificados por PEM em um arquivo chamado `CertificateChain.pem`.
 - Armazene a chave privada não criptografada codificada por PEM em um arquivo chamado `PrivateKey.pem`.
2. Para usar o exemplo de comando a seguir, substitua esses nomes de arquivos pelos nomes dos seus próprios arquivos e substitua '`regional.example.com cert`' por um nome para o certificado

do qual foi feito upload. Digite o comando em uma única linha contínua. O exemplo a seguir inclui quebras de linha e espaços extras para facilitar a leitura.

```
aws apigateway create-domain-name --domain-name 'regional.example.com'  
--regional-certificate-name 'regional.example.com cert'  
--certificate-body file://Certificate.pem  
--certificate-private-key file://PrivateKey.pem  
--certificate-chain file://CertificateChain.pem  
--endpoint-configuration types=REGIONAL
```

Se houver êxito, a chamada retornará um resultado semelhante ao seguinte:

```
{  
    "certificateUploadDate": "2017-10-13T23:02:54Z",  
    "domainName": "regional.example.com",  
    "endpointConfiguration": {  
        "types": "REGIONAL"  
    },  
    "regionalCertificateArn": "arn:aws:apigateway:us-west-2:123456789012:certificate/  
c19332f0-3be6-457f-a244-e03a423084e6",  
    "regionalDomainName": "d-numh1z56v6.execute-api.us-west-2.amazonaws.com"  
}
```

O valor da propriedade `regionalDomainName` 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 que está vinculado ao nome de domínio personalizado seja direcionado para o nome de host dessa API regional.

3. Adicione um mapeamento de caminho base para expor a API especificada (por exemplo, `0qzs2sy7bh`) em uma etapa de implantação (por exemplo, `test`) com o nome de domínio personalizado especificado (por exemplo, `regional.example.com`).

```
aws apigateway create-base-path-mapping \  
--domain-name 'regional.example.com' \  
--base-path 'RegionalApiTest' \  
--rest-api-id 0qzs2sy7bh \  
--stage 'test'
```

Como resultado, a URL base usando o nome de domínio personalizado para a API que é implantada na etapa se torna `https://regional.example.com/RegionalApiTest`.

4. Note

Esta etapa pressupõe que seu domínio esteja hospedado no Route 53.

Configure os registros DNS para mapearem o nome de domínio personalizado regional ao nome de domínio de destino. Primeiro, crie o arquivo JSON que contém a definição para a configuração de um registro DNS para o nome de domínio regional. O exemplo a seguir mostra como criar um registro A DNS regional para mapear um nome de domínio personalizado (`regional.example.com`) ao seu 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 pegar os valores do nome de domínio de destino e do ID da zona hospedada, respectivamente, do nome de domínio personalizado que você vê no console do API Gateway. Você também pode obter os IDs de zona hospedada regional do Route 53 em [Regiões e endpoints do API Gateway](#).

```
{  
    "Changes": [  
        {
```

```
"Action": "CREATE",
"ResourceRecordSet": [
    {
        "Name": "regional.example.com",
        "Type": "A",
        "AliasTarget": {
            "DNSName": "d-numh1z56v6.execute-api.us-west-2.amazonaws.com",
            "HostedZoneId": "Z2OJLYMU09EFXC",
            "EvaluateTargetHealth": false
        }
    }
]
```

5. Execute o seguinte comando da CLI:

```
aws route53 change-resource-record-sets \
--hosted-zone-id {your-hosted-zone-id} \
--change-batch file://path/to/your/setup-dns-record.json
```

onde *{your-hosted-zone-id}* é o ID de zona hospedada do Route 53 do conjunto de registros DNS em sua conta. O valor do parâmetro *change-batch* aponta para um arquivo JSON (*setup-dns-record.json*) em uma pasta (*path/to/your*).

Migrar um nome de domínio personalizado para outro endpoint de API

Você pode migrar seu nome de domínio personalizado entre endpoints regionais e otimizados para fronteiras. 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. Uma última etapa opcional é remover os dados de configuração obsoletos do nome de domínio personalizado.

Ao planejar a migração, lembre-se de que para o nome de domínio personalizado da API otimizada para fronteiras, o certificado necessário fornecido pelo ACM deve ser da Leste dos EUA (Norte da Virgínia) Região (`us-east-1`). Esse certificado é distribuído por todas as localizações geográficas. No entanto, para uma API regional, o certificado do ACM para o nome de domínio regional deve ser da mesma região que hospeda a API. Você pode migrar um nome de domínio personalizado otimizado para fronteiras que não esteja na região `us-east-1` para um nome de domínio personalizado regional solicitando primeiro um novo certificado ACM da região que é local para a API.

Pode levar até 60 segundos para concluir uma migração entre um nome de domínio personalizado otimizado para fronteiras e um nome de domínio personalizado regional no API Gateway. Para o endpoint recém-criado ficar pronto para aceitar o tráfego, o tempo de migração também depende de quando você atualiza seus registros DNS.

Tópicos

- [Migrar nomes de domínios regionais e otimizados para fronteiras usando o console do API Gateway \(p. 698\)](#)
- [Migrar nomes de domínios personalizados usando a AWS CLI \(p. 698\)](#)

Migrar nomes de domínios regionais e otimizados para fronteiras usando o console do API Gateway

Para usar o console do API Gateway para migrar um nome de domínio personalizado regional para um nome de domínio personalizado otimizado para fronteiras e vice-versa, use o procedimento a seguir.

Para migrar nomes de domínios regionais e otimizados para fronteiras usando o console do API Gateway

1. Faça login no console do API Gateway e selecione Custom Domain Names (Nomes de domínio personalizados) no painel de navegação principal.
2. Escolha um nome de domínio existente de Nomes de domínios personalizados e, em seguida, escolha Editar.
3. Dependendo do tipo de endpoint existente, faça o seguinte:
 - a. Para um nome de domínio otimizado para fronteiras, escolha Adicionar configuração regional.
 - b. Para um nome de domínio regional, escolha Adicionar configuração para fronteiras.
4. Escolha um certificado na lista suspensa.
5. Escolha Salvar.
6. Escolha Prosseguir para confirmar a adição do novo endpoint.
7. Atualize os registros DNS para apontar o novo nome de domínio para o nome de domínio de destino recém-provisionado.

Migrar nomes de domínios personalizados usando a AWS CLI

Para usar a AWS CLI para atualizar um nome de domínio personalizado de um endpoint otimizado para fronteiras para um endpoint regional ou vice-versa, chame o comando `update-domain-name` para adicionar o novo tipo de endpoint e, opcionalmente, chame o comando `update-domain-name` para remover o antigo tipo de endpoint.

Tópicos

- [Migrar um nome de domínio personalizado otimizado para fronteiras para regional \(p. 698\)](#)
- [Migrar um nome de domínio personalizado regional para otimizado para fronteiras \(p. 699\)](#)

Migrar um nome de domínio personalizado otimizado para fronteiras para regional

Para migrar um nome de domínio personalizado otimizado para fronteiras para um nome de domínio personalizado regional, chame o comando `update-domain-name` da CLI da seguinte forma:

```
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 que a API regional.

A resposta bem-sucedida tem um código de status 200 OK e um corpo semelhante ao seguinte:

```
{
```

```
"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-fdisjghn6.execute-api.us-west-2.amazonaws.com"
}
```

Para o nome de domínio personalizado regional atualizado, 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 fronteiras chamando o comando `update-domain-name` da AWS CLI:

```
aws apigateway update-domain-name \
--domain-name api.example.com \
--patch-operations [ \
    {op:'remove', path:'/endpointConfiguration/types', value:'EDGE'}, \
    {op:'remove', path:'/certificateName'}, \
    {op:'remove', path:'/certificateArn'} \
]
```

Migrar um nome de domínio personalizado regional para otimizado para fronteiras

Para migrar um nome de domínio personalizado regional para um nome de domínio personalizado otimizado para fronteiras, chame o comando `update-domain-name` da AWS CLI, da seguinte forma:

```
aws apigateway update-domain-name \
--domain-name 'api.example.com' \
--patch-operations [ \
    { op:'add', path:'/endpointConfiguration/types',value: 'EDGE' }, \
    { op:'add', path:'/certificateName', value:'edge-cert'}, \
    { op:'add', path:'/certificateArn', value: 'arn:aws:acm:us-east-1:123456789012:certificate/34a95aa1-77fa-427c-aa07-3a88bd9f3c0a' } \
]
```

O certificado de domínio otimizado para fronteiras deve ser criado na região `us-east-1`.

A resposta bem-sucedida tem um código de status 200 `OK` e um corpo semelhante ao seguinte:

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

```
        "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 fronteiras 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 fronteiras seja roteado para o nome de host da API otimizada para fronteiras.

Depois que o registro DNS for definido, você poderá remover o tipo de endpoint `REGION` do nome de domínio personalizado:

```
aws apigateway update-domain-name \
--domain-name api.example.com \
--patch-operations [ \
    {op:'remove', path:'/endpointConfiguration/types', value:'REGIONAL'}, \
    {op:'remove', path:'regionalCertificateArn'} \
]
```

O resultado desse comando é semelhante à seguinte saída, com os dados da configuração do nome de domínio otimizado para fronteiras:

```
{
    "certificateArn": "arn:aws:acm:us-east-1:738575810317:certificate/34a95aa1-77fa-427c-
aa07-3a88bd9f3c0a",
    "certificateName": "edge-cert",
    "certificateUploadDate": "2017-10-16T23:22:57Z",
    "distributionDomainName": "d1frvgze7vy1bf.cloudfront.net",
    "domainName": "regional.haymuto.com",
    "endpointConfiguration": {
        "types": "EDGE"
    }
}
```

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 outro método, você pode exportá-la para um arquivo do OpenAPI usando o recurso Export API do API Gateway, que faz parte do Amazon API Gateway Control Service. Existem opções para incluir as extensões de integração do API Gateway, bem como as extensões [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 o recurso Export API do API Gateway, consulte [Fazendo solicitações HTTP](#).

Note

Se você definir modelos na sua API, eles deverão ser para o tipo de conteúdo "application/json" para o API Gateway exportar esses modelos. 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
```

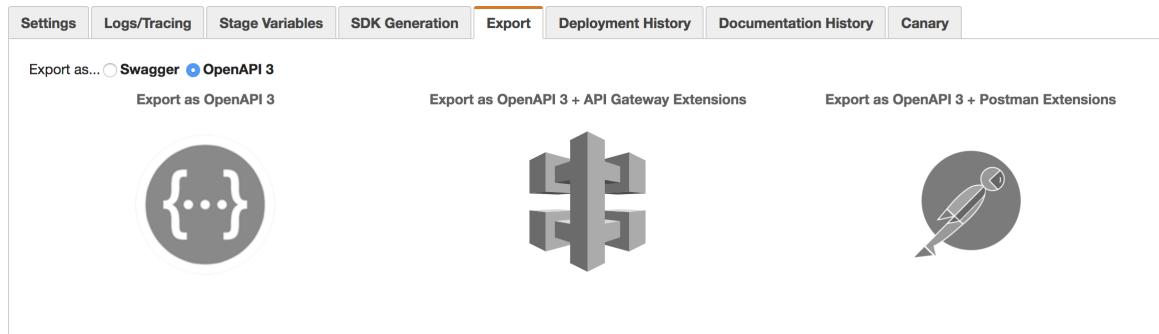
OpenAPI 2.0

```
GET /restapis/<restapi_id>/stages/<stage_name>/exports/swagger?extensions=integrations
Host: apigateway.<region>.amazonaws.com
Accept: application/yaml
```

Exportar a API REST usando o console do API Gateway

Depois de [implantar sua API REST em um estágio \(p. 518\)](#), você pode prosseguir e exportar essa API no estágio para um arquivo do OpenAPI usando o console do API Gateway.

Na página stage configuration (configuração do estágio) no console do API Gateway, selecione Export (Exportar) e, então, uma das opções disponíveis (Export as OpenAPI (Exportar como OpenAPI), Export as OpenAPI + API Gateway Integrations (Exportar como integrações OpenAPI + API Gateway) e Export as Postman (Exportar como Postman)), para fazer download da definição do OpenAPI da sua API.



Definir uma implantação da versão Canary no 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 como um versão canary 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 desempenho 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 \(p. 704\)](#)
- [Criar uma implantação da versão Canary \(p. 705\)](#)
- [Atualizar uma versão canary \(p. 709\)](#)
- [Promover uma versão canary \(p. 711\)](#)
- [Desativar uma versão canary \(p. 713\)](#)

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 inclusive, 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. Elas são relatados em um CloudWatch Logs do estágio de produção, grupo de registro, assim como grupo de registros do CloudWatch Logs específico do 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 revertem a versão canary do estágio de produção.

O grupo de registros de execução do estágio de produção é chamado `API-Gateway-Execution-Logs/{rest-api-id}/{stage-name}` e o grupo de registros 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 registros ou escolher um existente. O nome do grupo de registros de acesso da versão canary tem o sufixo `/Canary` anexado ao nome do grupo de registros 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

Você cria uma implantação da versão canary ao implantar a API as [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 já existente fazendo uma `stage:update` solicitação 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 nenhuma implantação da versão canary.

Você pode criar uma implantação da versão canary no API Gateway usando o console do API Gateway, a AWS CLI ou um AWS SDK.

Tópicos

- [Criar uma implantação do canary usando o console do API Gateway \(p. 705\)](#)
- [Criar uma implantação do canary usando a AWS CLI \(p. 707\)](#)

Criar uma implantação do 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. Faça login no console do API Gateway.
2. Escolha uma API existente ou crie uma nova API.
3. Altere a API, se necessário, ou configure os métodos e integrações de API desejados.
4. Escolha Deploy API (Implantar API) no menu suspenso Actions (Ações). 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.

5. Para ativar o armazenamento em cache da API, escolha a guia Settings no Stage Editor e siga as instruções na tela. Para obter mais informações, consulte [the section called “Habilitar o armazenamento em cache de APIs” \(p. 525\)](#).
6. Para definir variáveis de estágio, escolha a guia Stage Variables em Stage Editor e siga as instruções na tela para adicionar ou modificar as variáveis de estágio. Para obter mais informações, consulte [the section called “Configurar variáveis de estágio para uma API REST” \(p. 540\)](#).
7. Para configurar a execução ou registro de acesso, escolha a guia Logs em Stage Editor e siga as instruções na tela. Para obter mais informações, consulte [Configurar registro de API em logs do CloudWatch no API Gateway \(p. 536\)](#).
8. Em Stage Editor, escolha a guia Canary e, depois, escolha Create Canary.
9. Na seção Stage's Request Distribution, escolha o ícone de lápis ao lado de Percentage of requests to Canary e digite um número (por exemplo, 5 . 0) no campo de texto de entrada. Escolha o ícone de marca de seleção para salvar a configuração.
10. Para associar uma ACL da web do AWS WAF com o estágio, selecione uma ACL da web da lista suspensa de ACL da web.

Note

Se necessário, selecione Criar ACL da web para abrir o console do AWS WAF em uma nova guia do navegador, crie a ACL da web e retorne ao console do API Gateway para associar a ACL da web ao estágio.

11. Se desejar, selecione Bloquear solicitações de API se não for possível avaliar a WebACL (Falha - Fechar).
12. Se necessário, escolha Add Stage Variables para adicioná-las a seção Canary Stage Variables para sobreescriver as variáveis de estágio existentes ou adicionar novas variáveis de estágio à versão canary.
13. Se desejar, escolha Enable use of stage cache para ativar o armazenamento em cache para a versão canary e salvar a sua escolha. O cache não estará disponível para a versão canary até o armazenamento em cache da API ser ativado.

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, escolha Deploy API no menu suspenso Actions ao lado da lista Resources.
2. Em Deploy API, escolha o estágio ativado no momento para canary na lista suspensa Deployment stage.
3. Opcionalmente, digite uma descrição em Deployment description.
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 \(p. 706\)](#).

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 tem um novo valor deploymentId, enquanto o estágio ainda mantém o valor deploymentId inicial. Enquanto isso, o console chama `stage:update`.

Criar uma implantação do canary usando a AWS CLI

Primeiro, crie uma implantação de linha de base com duas variáveis de estágio, mas sem qualquer canary:

```
aws apigateway create-deployment  
--variables sv0=val0,sv1=val1  
--rest-api-id 4wk1k4onj3  
--stage-name prod
```

O comando retorna uma representação do [Deployment](#) resultante, similar ao seguinte:

```
{  
    "id": "du4ot1",  
    "createdDate": 1511379050  
}
```

O `id` da implantação resultante identifica um snapshot (ou versão) da API.

Agora, crie uma implantação do canary no estágio `prod`:

```
aws apigateway create-deployment  
--canary-settings '{ \  
    "percentTraffic":10.5, \  
    "useStageCache":false, \  
    "stageVariableOverrides":{ \  
        "sv1":"val2", \  
        "sv2":"val3" \  
    } \  
}' \  
--rest-api-id 4wk1k4onj3 \  
--stage-name prod
```

Se o estágio especificado (`prod`) não existir, o comando anterior retornará um erro. Caso contrário, ele retornará a representação do recurso [deployment](#) recém-criada semelhante à seguinte:

```
{  
    "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. Você pode visualizar esta representação do estágio chamando o comando `get-stage`, similar ao seguinte:

```
aws apigateway get-stage --rest-api-id 4wk1k4onj3 --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 pode criar uma implantação da versão canary a partir de uma implantação normal, atualizando o estágio para ativar um canary. Para demonstrar isso, crie uma implantação normal primeiro:

```
aws apigateway create-deployment \
--variables sv0=val0,sv1=val1 \
--rest-api-id 4wk1k4onj3 \
--stage-name beta
```

O comando retorna uma representação de implantação da versão base:

```
{
    "id": "cifeiw",
    "createdDate": 1511380879
}
```

O estágio beta associado não possui configurações do canary:

```
{
    "stageName": "beta",
    "variables": {
        "sv0": "val0",
        "sv1": "val1"
    },
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "deploymentId": "cifeiw",
    "lastUpdatedDate": 1511380879,
    "createdDate": 1511380879,
    "methodSettings": {}
}
```

Agora, crie uma nova nova implantação da versão canary anexando um canary ao estágio:

```
aws apigateway update-stage \
```

```
--patch-operations '[{ \
    "op":"replace", \
    "path":"/canarySettings/percentTraffic", \
    "value":"10.5" \
},{ \
    "op":"replace", \
    "path":"/canarySettings/useStageCache", \
    "value":"false" \
},{ \
    "op":"replace", \
    "path":"/canarySettings/stageVariableOverrides/sv1", \
    "value":"val2" \
},{ \
    "op":"replace", \
    "path":"/canarySettings/stageVariableOverrides/sv2", \
    "value":"val3" \
}]' \
--rest-api-id 4wk1k4onj3 \
--stage-name beta
```

Uma representação de um estágio atualizado é semelhante ao seguinte:

```
{
  "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 ativar um canary em uma versão existente da API, a versão de produção (`Stage`) e a versão canary (`canarySettings`) apontam para a mesma implantação, ou seja, a mesma versão (`deploymentId`) da API. 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 o desempenho. 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 uma versão canary usando o console do API Gateway, o comando da AWS CLI `update-stage` ou um AWS SDK.

Tópicos

- [Atualizar uma versão canary usando o console do API Gateway \(p. 710\)](#)
- [Atualizar uma versão canary usando a AWS CLI \(p. 710\)](#)

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:

1. Faça login no console do API Gateway e escolha uma API existente no painel de navegação principal.
2. Escolha Stages na API e, depois, escolha o estágio existente na lista Stages para abrir o Stage Editor.
3. Escolha a guia Canary em Stage Editor.
4. Atualize Percentage of requests directed to Canary aumentando ou diminuindo o número percentual entre 0,0 e 100,0, inclusive.
5. Atualize Canary Stage Variables, incluindo a adição, a remoção ou a modificação de uma variável de estágio desejada.
6. Atualize a opção Enable use of stage cache selecionando ou limpando a caixa de seleção.
7. Salve as alterações.

Atualizar uma versão canary usando a AWS CLI

Para usar a AWS CLI para atualizar um canary, chame o comando `update-stage`.

Para ativar ou desativar o uso de um cache do estágio para o canary, chame o comando `update-stage`, da seguinte forma:

```
aws apigateway update-stage \
  --rest-api-id {rest-api-id} \
  --stage-name '{stage-name}' \
  --patch-operations op=replace,path=/canarySettings/useStageCache,value=true
```

Para ajustar a porcentagem do tráfego do canary, chame `update-stage` para substituir o valor / `canarySettings/percentTraffic` no **estágio**.

```
aws apigateway update-stage \
  --rest-api-id {rest-api-id} \
  --stage-name '{stage-name}' \
  --patch-operations op=replace,path=/canarySettings/percentTraffic,value=25.0
```

Para atualizar as variáveis de estágio do canary, incluindo a adição, a substituição ou a remoção de uma variável de estágio do canary:

```
aws apigateway update-stage \
  --rest-api-id {rest-api-id} \
  --stage-name '{stage-name}' \
  --patch-operations '[{
    "op": "replace",
    "path": "/canarySettings/stageVariableOverides/newVar",
    "value": "newVal"
  }, {
    "op": "replace",
    "path": "/canarySettings/stageVariableOverides/var2"
  }]
```

```
        "value": "val4", \
    }, {
        "op": "remove",
        "path": "/canarySettings/stageVariableOverrides/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": "/canary/percentTraffic",
    "value": "20.0"
}, {
    "op": "replace",
    "path": "/canary/useStageCache",
    "value": "true"
}, {
    "op": "remove",
    "path": "/canarySettings/stageVariableOverrides/var1"
}, {
    "op": "replace",
    "path": "/canarySettings/stageVariableOverrides/newVar",
    "value": "newVal"
}, {
    "op": "replace",
    "path": "/canarySettings/stageVariableOverrides/val2",
    "value": "val4"
}]'
```

Promover uma versão canary

Para promover uma versão canary, disponibilize no estágio de produção a versão da API em teste. A operação envolve as seguintes tarefas:

- Redefina o [deployment ID](#) do estágio com as configurações do [deployment ID](#) 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 %.

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.

Tópicos

- [Promover uma versão canary usando o console do API Gateway \(p. 711\)](#)
- [Promover uma versão canary usando a AWS CLI \(p. 712\)](#)

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:

1. Faça login no console do API Gateway e escolha uma API existente no painel de navegação principal.
2. Escolha Stages na API e, depois, escolha o estágio existente na lista Stages para abrir o Stage Editor.
3. Escolha a guia Canary em Stage Editor.
4. Escolha Promote Canary.
5. Confirme as alterações a serem realizadas e escolha Update.

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" \(p. 712\)](#).

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": "nfcn0x",  
    "lastUpdatedDate": "2017-11-22T00:54:28Z",  
    "methodSettings": {  
        ...  
    },  
    "stageName": "prod",  
    "variables": {  
        "sv1": "val1"  
    }  
}
```

Chamamos a seguinte solicitação `update-stage` para promovê-la:

```
aws apigateway update-stage  
  --rest-api-id '{rest-api-id}'  
  --stage-name '{stage-name}'  
  --patch-operations '[{  
      "op": "replace",  
      "value": "0.0"  
      "path": "/canarySettings/percentTraffic",  
  }]
```

```
}, {
    "op": "copy",
    "from": "/canarySettings/stageVariableOverrides",
    "path": "/variables",
},
{
    "op": "copy",
    "from": "/canarySettings/deploymentId",
    "path": "/deploymentId"
} ]'
```

Após a promoção, o estágio agora se parece com:

```
{
    "_links": {
        ...
    },
    "accessLogSettings": {
        ...
    },
    "cacheClusterEnabled": false,
    "cacheClusterStatus": "NOT_AVAILABLE",
    "canarySettings": {
        "deploymentId": "eh1sby",
        "useStageCache": false,
        "stageVariableOverrides": {
            "sv2": "val3",
            "sv1": "val2"
        },
        "percentTraffic": 0
    },
    "createdDate": "2017-11-20T04:42:19Z",
    "deploymentId": "eh1sby",
    "lastUpdatedDate": "2017-11-22T05:29:47Z",
    "methodSettings": {
        ...
    },
    "stageName": "prod",
    "variables": {
        "sv2": "val3",
        "sv1": "val2"
    }
}
```

Como você pode ver, promover a versão canary para o estágio não desabilita o canary e a implantação continua sendo uma implantação da versão canary. 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 canary” \(p. 713\)](#).

Desativar uma versão canary

Para desativar uma implantação da versão canary, defina `canarySettings` como nulo para removê-la do estágio.

Você pode desabilitar uma implantação da versão canary usando o console do API Gateway, a AWS CLI ou um AWS SDK.

Tópicos

- [Desabilitar uma versão canary usando o console do API Gateway \(p. 714\)](#)
- [Desabilitar uma versão canary usando a AWS CLI \(p. 714\)](#)

Desabilitar uma versão canary usando o console do API Gateway

Para usar o console do API Gateway para desativar uma implantação da versão canary, siga as seguintes etapas:

1. Faça login no console do API Gateway e escolha uma API existente no painel de navegação principal.
2. Escolha Stages na API e, depois, escolha o estágio existente na lista Stages para abrir o Stage Editor.
3. Escolha a guia Canary em Stage Editor.
4. Escolha Delete Canary.
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 “Desabilitar uma versão canary usando a AWS CLI” \(p. 714\)](#).

Desabilitar uma versão canary usando a AWS CLI

Para usar a AWS CLI para desativar uma implantação da versão canary, chame o comando `update-stage`, da seguinte maneira:

```
aws apigateway update-stage \
--rest-api-id 4wk1k4onj3 \
--stage-name canary \
--patch-operations '[{"op":"remove", "path":"/canarySettings"}]
```

Uma resposta bem-sucedida retorna uma carga similar à seguinte:

```
{
  "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 canary.

Monitorar a configuração de API do API Gateway com o 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 alterações de configurações para recursos do API Gateway é útil para casos de uso de solução de problemas operacionais, de auditoria e de conformidade.

O AWS Config 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 para o Amazon API Gateway no AWS Config está disponível em todas as regiões públicas da AWS e em AWS GovCloud (US). Para obter uma lista completa de regiões com suporte, consulte [Regiões e endpoints](#) no AWS General Reference.

Tópicos

- [Tipos de recursos com suporte \(p. 715\)](#)
- [Configuração de AWS Config \(p. 716\)](#)
- [Configuração do AWS Config para registrar recursos do API Gateway \(p. 716\)](#)
- [Visualizar detalhes de configuração do API Gateway no console do AWS Config \(p. 716\)](#)
- [Avaliar recursos do API Gateway usando regras do AWS Config \(p. 717\)](#)

Tipos de recursos com suporte

Os tipos de recursos do API Gateway a seguir são integrados ao AWS Config e estão documentados em [Relacionamentos de recursos e tipos de recursos da AWS com suporte do AWS Config](#):

- `AWS::ApiGatewayV2::Api` (API WebSocket)

- `AWS::ApiGateway::RestApi` (API REST)
- `AWS::ApiGatewayV2::Stage` (Estágio de API WebSocket)
- `AWS::ApiGateway::Stage` (Estágio de API REST)

Para obter mais informações sobre o AWS Config, consulte [AWS Config Developer Guide](#). 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 \(p. 516\)](#) 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`

Configuração de AWS Config

Para configurar o AWS Config inicialmente, consulte os seguintes tópicos no [AWS Config Developer Guide](#).

- [Configuração do AWS Config com o 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](#).

Visualizar detalhes de configuração do API Gateway no console do AWS Config

Você pode usar o console do AWS Config para procurar recursos do API Gateway e obter detalhes históricos e atuais sobre suas configurações. O seguinte procedimento mostra como encontrar informações sobre uma API do API Gateway.

Para encontrar um recurso do API Gateway no console do AWS Config

1. Abra o [console do AWS Config](#).

2. Selecione 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 encontrar um recurso e visualizar informações desta página, consulte [Exibir histórico e configurações de recursos da AWS](#) no AWS Config Developer Guide.

Avaliar recursos do API Gateway usando regras do AWS Config

Você pode criar regras do AWS Config, que representam suas definições ideais de configuração 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 viola uma regra e é sinalizado como não compatível, o AWS Config pode alertá-lo usando um tópico do [Guia do desenvolvedor do Amazon Simple Notification Service](#) (Amazon SNS). Para consumir de forma programática os dados nesses alertas do AWS Config, use uma fila do Amazon Simple Queue Service (Amazon SQS) como endpoint de notificação para o tópico do Amazon SNS.

Para saber mais sobre como configurar e usar regras, consulte [Avaliar recursos com regras](#) no AWS Config Developer Guide.

Marcar seus 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 ativa essas tags no painel do AWS Billing and Cost Management. A AWS usa as tags para categorizar seus custos e fornecer um relatório mensal de alocação de custos mensais a você. Para obter mais informações, consulte [Usar tags de alocação de custos](#) no Guia do usuário do AWS Billing and Cost Management.
- Identifique e organize 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 Eventos do CloudWatch.

Para obter dicas sobre como usar tags, consulte a postagem [Estratégias de marcação da AWS](#) no blog AWS Answers.

As próximas seções contêm mais informações sobre tags para o Amazon API Gateway.

Tópicos

- [Recursos do API Gateway que podem ser marcados \(p. 718\)](#)
- [Usar tags para controlar o acesso aos recursos do API Gateway \(p. 720\)](#)

Recursos do API Gateway que podem ser marcados

Tags podem ser definidas nos seguintes recursos da API do WebSocket na [API V2 do Amazon API Gateway](#):

- `Api`
- `DomainName`

- Stage

Além disso, tags podem ser definidas nos seguintes recursos da API REST na [API V1 do Amazon API Gateway](#):

- ApiKey
- ClientCertificate
- DomainName
- RestApi
- Stage
- UsagePlan
- VpcLink

As tags não podem ser definidas diretamente em outros recursos. No entanto, na [API V1 do Amazon API Gateway](#), os recursos filho herdam as tags que são definidas nos recursos pai. Por exemplo:

- Se uma tag for definida em um recurso RestApi, essa tag será herdada pelos seguintes recursos filho do RestApi:
 - Authorizer
 - Deployment
 - Documentation
 - GatewayResponse
 - Integration
 - Method
 - Model
 - Resource
 - ResourcePolicy
 - Setting
 - Stage
- Se uma tag for definida em um DomainName, essa tag será herdada por todos os recursos BasePathMapping sob ela.
- Se uma tag for definida em um UsagePlan, essa tag será herdada por todos os recursos UsagePlanKey sob ela.

Herança de tags na API V1 do Amazon API Gateway

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 RestApi pai.

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 seguintes restrições e convenções de uso 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 tamanho máximo do valor da tag é de 256 caracteres Unicode em UTF-8.
- Os caracteres permitidos para chaves e valores são letras, números e espaços representáveis em UTF-8, e os seguintes caracteres: . : + = @ _ / - (hífen). Os recursos do Amazon EC2 permitem quaisquer caracteres.
- 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. Não é possível editar nem excluir chaves nem valores de tag com esse prefixo. As tags com esse prefixo não contam em relação às tags por limite de recurso.

Usar tags para controlar o acesso aos recursos 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” \(p. 378\)](#). 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.

Usar o controle de acesso baseado em tags pode permitir um controle mais preciso do que o controle no nível de API, bem como um controle mais dinâmico do que o controle de acesso baseado em recursos. As

políticas do IAM podem ser criadas para permitir ou não uma operação com base em tags fornecidas na solicitação (tags de solicitação) ou tags no recurso que está sendo operado (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.

Exemplo 1: limitar ações com base em tags de recursos

O exemplo de política a seguir concede permissão aos usuários para executar ações GET em todos os recursos. Além disso, se um recurso tiver uma tag chamada `iamrole` com um valor `readWrite`, a política concederá aos usuários permissão para executar todas as ações no recurso.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "apigateway:GET",  
            "Resource": "*"  
        },  
        {  
            "Effect": "Allow",  
            "Action": "apigateway:*",  
            "Resource": "*",  
            "Condition": {  
                "StringEquals": {  
                    "aws:ResourceTag/iamrole": "readWrite"  
                }  
            }  
        }  
    ]  
}
```

Exemplo 2: limitar ações com base em tags na solicitação

A política de exemplo a seguir especifica que:

- Quando o usuário criar um novo estágio, a solicitação para criar o estágio deverá conter uma tag chamada `stage`.
- O valor da tag `stage` deverá ser `beta`, `gamma` ou `prod`. Caso contrário, a solicitação para criar o estágio será negada.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "apigateway:PUT/  
            ]  
        }  
    ]  
}
```

```
        "apigateway:*"
    ],
    "Resource": [
        "*"
    ]
},
{
    "Effect": "Deny",
    "Action": "apigateway:POST",
    "Resource": "arn:aws:apigateway:*/:::restapis/*/*stages",
    "Condition": {
        "Null": {
            "aws:RequestTag/stage": "true"
        }
    }
},
{
    "Effect": "Deny",
    "Action": "apigateway:POST",
    "Resource": "arn:aws:apigateway:*/:::restapis/*/*stages",
    "Condition": {
        "ForAnyValue:StringNotEquals": {
            "aws:RequestTag/stage": [
                "beta",
                "gamma",
                "prod"
            ]
        }
    }
}
]
```

Exemplo 3: negar ações com base em tags de recurso

A política de exemplo a seguir permite que os usuários executem todas as ações nos recursos do API Gateway por padrão. Se um recurso tiver uma tag chamada `stage` com um valor de `prod`, os usuários não terão permissão para realizar modificações (`PATCH`, `PUT`, `POST`, `DELETE`) no recurso.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "apigateway:*",
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action": [
                "apigateway:PATCH",
                "apigateway:PUT",
                "apigateway:POST",
                "apigateway:DELETE"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/stage": "prod"
                }
            }
        }
    ]
}
```

```
}
```

Exemplo 4: permitir ações com base em tags de recursos

Este exemplo de política permite que os usuários executem todas as ações em todos os recursos do API Gateway por padrão. Se um recurso tiver uma tag chamada `environment` cujo valor seja `prod`, os usuários não terão permissão para executar nenhuma operação no recurso.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "apigateway:*",
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action": [
                "apigateway:*
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/environment": "prod"
                }
            }
        }
    ]
}
```

Referências de API na Versão 1 e Versão 2 do Amazon API Gateway

O Amazon API Gateway fornece APIs para criar e implantar seus próprios HTTP e APIs WebSocket. Além disso, as APIs do API Gateway estão disponíveis nos SDKs padrão da AWS.

Se estiver usando um idioma para o qual exista um SDK da AWS, você pode preferir 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 onde encontrar a documentação de referência de SDKs da AWS e API REST do API Gateway:

- [Ferramentas para Amazon Web Services](#) (para ferramentas do desenvolvedor, SDKs e toolkits de IDE)
- [Referência de API na Versão 1 do Amazon API Gateway](#) (para criar e implantar APIs HTTP)
- [Referência de API na Versão 2 do Amazon API Gateway](#) (para criar e implantar APIs WebSocket)

Limites do Amazon API Gateway e observações importantes

Tópicos

- [Limites do API Gateway \(p. 725\)](#)
- [Observações importantes do Amazon API Gateway \(p. 730\)](#)

Limites do API Gateway

Salvo indicação em contrário, os limites podem ser aumentados mediante solicitação. Para solicitar um aumento de limite, é possível usar o [Cotas de serviço](#) ou entrar em contato com o [AWS Support Center](#).

Quando a autorização está habilitada em um método, o comprimento máximo do ARN desse método (por exemplo, `arn:aws:execute-api:{region-id}:{account-id}:{api-id}/{stage-id}/{method}/{resource}/{path}`) 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.

Note

Isso limita o tamanho do URI quando as políticas de recurso são usadas. No caso de APIs privadas em que uma política de recurso é necessária, isso limita o tamanho do URI de todas as APIs privadas.

Limites do nível da conta do API Gateway

Os limites a seguir aplicam-se ao nível da conta do Amazon API Gateway.

Recurso ou operação	Limite padrão	Pode ser aumentado
Controlar o limite por região em APIs REST, APIs WebSocket e APIs de retorno de chamada do WebSocket	10.000 solicitações por segundo (RPS) com uma capacidade de intermitência adicional fornecida pelo algoritmo de bucket de tokens , usando uma capacidade máxima de bucket de 5.000 solicitações. Note O limite de intermitência é determinado pela equipe de serviço do API Gateway com base nos limites de RPS geral da conta. O cliente não pode controlar ou solicitar alterações para esse limite.	Sim
APIs regionais	600	Não
APIs otimizadas para fronteiras	120	Não

Limites do API Gateway para configurar e executar uma API WebSocket

Os seguintes limites aplicam-se à configuração e execução de uma API WebSocket no Amazon API Gateway.

Recurso ou operação	Limite padrão	Pode ser aumentado
Novas conexões por segundo e por conta (em todas as APIs WebSocket) por região	500	Sim
Autorizadores do AWS Lambda por API	10	Sim
Tamanho do resultado do autorizador do AWS Lambda	8 KB	Não
Rotas por API	300	Sim
Integrações por API	300	Sim
Estágios por API	10	Sim
Tamanho de quadros WebSocket	32 KB	Não
Tamanho da carga da mensagem	128 KB Note Devido ao limite de tamanho de quadro WebSocket de 32 KB, uma mensagem maior do que 32 KB deve ser dividida em vários quadros, cada um contendo 32 KB ou menos. Se uma mensagem maior (ou quadro de tamanho maior) for recebida, a conexão será encerrada com o código 1009.	Não
Duração da conexão para API WebSocket	2 horas	Não
Tempo limite de inatividade	10 minutos	Não

Limites do API Gateway para configurar e executar uma API REST

Os seguintes limites aplicam-se à configuração e execução de uma API REST no Amazon API Gateway.

Recurso ou operação	Limite padrão	Pode ser aumentado
Nomes de domínio personalizados por conta por região	120	Sim
Comprimento, em caracteres, do URL de uma API otimizada para fronteiras	8192	Não
Comprimento, em caracteres, do URL de uma API regional	10240	Não
APIs privadas por conta por região	600	Não
Comprimento, em caracteres, da política de recursos do API Gateway	8192	Sim
Chaves de API por conta por região	500	Sim
Certificados de cliente por conta por região	60	Sim
Autorizadores por API (AWS Lambda e Amazon Cognito)	10	Sim
Partes da documentação por API	2000	Sim
Recursos por API	300	Sim
Estágios por API	10	Sim
Variáveis de estágio por estágio	100	Não
Comprimento, em caracteres, da chave em uma variável de estágio	64	Não
Comprimento, em caracteres, do valor em uma variável de estágio	512	Não
Planos de uso por conta por região	300	Sim

Recurso ou operação	Limite padrão	Pode ser aumentado
Planos de uso por chave de API	10	Sim
Links de VPC por conta por região	20	Sim
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	50 milissegundos a 29 segundos para todos os tipos de integração, incluindo integrações do Lambda, proxy do Lambda, HTTP, proxy HTTP e da AWS.	Não é destinado a limites inferiores ou superiores.
Tamanho total combinado de todos os valores do cabeçalho	10240 bytes	Não
Tamanho da carga	10 MB	Não
Tags por estágio	50	Não
Número de iterações em um loop <code>#foreach ... #end</code> em modelos de mapeamento	1000	Não
Comprimento do ARN de um método com autorização	1600 bytes	Não

Para `restapi:import` ou `restapi:put`, o tamanho máximo do arquivo de definição da API é 2 MB.

Todos os limites por API só podem ser aumentados em APIs específicas.

Limites do API Gateway para criação, implantação e gerenciamento de uma API

Os seguintes limites fixos aplicam-se à criação, à implantação e ao gerenciamento de uma API no API Gateway, usando a AWS CLI, o console do API Gateway ou a API REST do API Gateway e seus SDKs. Esses limites não podem ser aumentados.

Ação	Limite padrão	Pode ser aumentado
CreateApiKey	5 solicitações por segundo por conta	Não
CreateDeployment	1 solicitação a cada 5 segundos por conta	Não
CreateDocumentationVersion	1 solicitação a cada 20 segundos por conta	Não
CreateDomainName	1 solicitação a cada 30 segundos por conta	Não
CreateResource	5 solicitações por segundo por conta	Não
CreateRestApi	API regional ou privada <ul style="list-style-type: none"> • 1 solicitação a cada 3 segundos por conta API otimizada para fronteiras <ul style="list-style-type: none"> • 1 solicitação a cada 30 segundos por conta 	Não
DeleteApiKey	5 solicitações por segundo por conta	Não
DeleteDomainName	1 solicitação a cada 30 segundos por conta	Não
DeleteResource	5 solicitações por segundo por conta	Não
DeleteRestApi	1 solicitação a cada 30 segundos por conta	Não
GetResources	5 solicitações a cada 2 segundos por conta	Não
ImportDocumentationParts	1 solicitação a cada 30 segundos por conta	Não
ImportRestApi	API regional ou privada <ul style="list-style-type: none"> • 1 solicitação a cada 3 segundos por conta API otimizada para fronteiras <ul style="list-style-type: none"> • 1 solicitação a cada 30 segundos por conta 	Não
PutRestApi	1 solicitação por segundo por conta	Não

Ação	Limite padrão	Pode ser aumentado
UpdateAccount	1 solicitação a cada 20 segundos por conta	Não
UpdateDomainName	1 solicitação a cada 30 segundos por conta	Não
UpdateUsagePlan	1 solicitação a cada 20 segundos por conta	Não
Outras operações	Não há limite até o limite total da conta.	Não
Totais de operações	10 solicitações por segundo com um limite de intermitência de 40 solicitações por segundo.	Não

Observações importantes do Amazon API Gateway

Tópicos

- [Observações importantes do Amazon API Gateway para APIs REST e WebSocket \(p. 730\)](#)
- [Observações importantes do Amazon API Gateway para APIs WebSocket \(p. 731\)](#)
- [Observações importantes do Amazon API Gateway para APIs REST \(p. 731\)](#)

Observações importantes do Amazon API Gateway para APIs REST e WebSocket

- O API Gateway não oferece suporte ao compartilhamento de um nome de domínio personalizado em APIs REST e WebSocket.
- Nomes de estágio podem conter apenas caracteres alfanuméricos, hífens 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 1024 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.

Observações importantes do Amazon API Gateway para APIs WebSocket

- O API Gateway oferece suporte a 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.

Observações importantes do Amazon API Gateway para APIs REST

- O caractere pipe de texto simples (|) não tem suporte para nenhuma solicitação de sequência de consulta de URL e deve ser codificado pela URL.
- O caractere ponto-e-vírgula (;) não tem suporte para nenhuma string de consulta de URL da solicitação e resulta nos dados divididos.
- Ao usar o console do API Gateway para testar uma API, você pode obter uma resposta de "erros de endpoint desconhecido" se um certificado autoassinado for apresentado ao back-end, o certificado intermediário estiver ausente da cadeia de certificados ou qualquer outra exceção relacionada a certificados irreconhecíveis for lançada pelo back-end.
- Para uma entidade de [Resource](#) ou [Method](#) da API com integração privada, você deve 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 back-ends a seguir podem não oferecer suporte à autenticação de cliente SSL de uma forma que seja compatível com o API Gateway:
 - [NGINX](#)
 - [Heroku](#)
- O API Gateway oferece suporte à maior parte da [especificação OpenAPI 2.0](#) e a [especificação OpenAPI 3.0](#), com as seguintes exceções:
 - Segmentos de caminho só podem conter caracteres alfanuméricos, hifens, pontos, vírgula 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 têm suporte por meio de [autorizadores do Lambda \(p. 396\)](#). A configuração do OpenAPI é obtida por meio de [extensões do fornecedor \(p. 609\)](#).
 - O campo `deprecated` não tem suporte e é descartado em APIs exportadas.
 - Os modelos do API Gateway são definidos usando o [esquema JSON rascunho 4](#) em vez do esquema JSON usado pelo OpenAPI.
 - Os campos `additionalProperties` e `anyOf` não têm suporte em modelos.
 - O parâmetro `discriminator` não tem suporte em nenhum objeto de esquema.
 - Não há suporte para a tag `example`.
 - `exclusiveMinimum` não tem suporte no 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.

- `oneOf` não tem suporte no API Gateway.
- Não há suporte para o campo `readOnly`.
- 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 OpenAPI. Portanto, é necessário definir a segurança em um nível de operação a ser adequadamente aplicado.
- O API Gateway impõe as seguintes restrições e limitações ao lidar com métodos com a integração do Lambda ou a integração de 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 (<code>http/http_proxy/lambda</code>)	Resposta (<code>http/http_proxy/lambda</code>)
<code>Age</code>	Passagem	Passagem
<code>Accept</code>	Passagem	Descartado/ Passagem/ Passagem
<code>Accept-Charset</code>	Passagem	Passagem

Nome do cabeçalho	Solicitação (http/http_proxy/lambda)	Resposta (http/http_proxy/lambda)
Accept-Encoding	Passagem	Passagem
Authorization	Passagem	Remapped
Connection	Descartado/Passagem/Descartado	Remapped
Content-Encoding	Passagem/Descartado/Passagem	Passagem
Content-Length	Passagem (gerada com base no corpo)	Passagem
Content-MD5	Descartado	Remapped
Content-Type	Passagem	Passagem
Date	Passagem	Remapeado substituído
Expect	Descartado	Descartado
Host	5XX/5XX/Substituído pelo Lambda	Descartado
Max-Forwards	Descartado	Remapped
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
User-Agent	Passagem	Remapped
Via	Descartado/Descartado/Passagem	Passagem/ Descartado/ Descartado
Warn	Passagem	Passagem
WWW-Authenticate	Descartado	Remapped

- 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 oferece suporte a 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 back-end, 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 de `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 `` 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.

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.

- Última atualização da documentação: 21 de outubro de 2019

update-history-change	update-history-description	update-history-date
Nomes de domínio personalizados curinga (p. 735)	Adição de suporte para nomes de domínio personalizados curinga. Para obter mais informações, consulte Nomes de domínio personalizados curinga .	October 21, 2019
Registro em log do Amazon Kinesis Data Firehose (p. 735)	Adição de suporte ao Amazon Kinesis Data Firehose como um destino para dados de registro de acesso em logs. Para obter mais informações, consulte Usar o Amazon Kinesis Data Firehose como um destino para registro de acesso em logs ao API Gateway .	October 15, 2019
Aliases do Route53 para invocar APIs privadas (p. 735)	Adição de suporte para registros DNS de alias do Route53 adicionais para invocar APIs privadas. Para obter mais informações, consulte Acessar sua API privada usando um alias do Route53 .	September 18, 2019
Controle de acesso baseado em tags para APIs do WebSocket (p. 735)	Adição de suporte ao controle de acesso baseado em tags para APIs do WebSocket. Para obter mais informações, consulte Recursos do API Gateway que podem ser marcados .	June 27, 2019
Seleção de versão do TLS para domínios personalizados (p. 735)	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 .	June 20, 2019
Políticas de VPC endpoint para APIs privadas (p. 735)	Adição de suporte para melhorar a segurança de APIs privadas anexando políticas de endpoint a VPC endpoints de interface. Para obter mais informações,	June 4, 2019

	consulte Usar políticas de VPC endpoint para APIs privadas no API Gateway .	
Documentação atualizada (p. 735)	Os Conceitos básicos do Amazon API Gateway foram reescritos. Tutoriais movidos para Tutoriais do Amazon API Gateway .	May 29, 2019
Controle de acesso baseado em tags para APIs REST (p. 735)	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 a recursos do API Gateway .	May 23, 2019
Documentação atualizada (p. 735)	6 tópicos reescritos: O que é o Amazon API Gateway? , TUTORIAL: Criar uma API com integração de proxy HTTP , TUTORIAL: Criar uma API REST Calc com três integrações não proxy , Referência de variáveis de registro em log de acesso e modelo de mapeamento do API Gateway , Uso de autorizadores Lambda do API Gateway e Habilitar o CORS para um recurso da API REST do API Gateway .	April 5, 2019
Melhorias no portal do desenvolvedor sem servidor (p. 735)	Adição 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 Uso de um portal do desenvolvedor para catalogar as suas APIs .	March 28, 2019
Suporte para AWS Config (p. 735)	Adicionado o suporte para AWS Config. Para obter mais informações, consulte Monitoramento da API do API Gateway com o AWS Config .	March 20, 2019
Suporte para AWS CloudFormation (p. 735)	Adição da API V2 do API Gateway à referência de modelo do AWS CloudFormation. Para obter mais informações, consulte Referência de tipos de recurso V2 do Amazon API Gateway .	February 7, 2019

Suporte para APIs WebSocket (p. 735)	Suporte adicionado para APIs WebSocket. Para obter mais informações, consulte Criação de uma API WebSocket no Amazon API Gateway .	December 18, 2018
Portal do desenvolvedor sem servidor disponível por meio do AWS Serverless Application Repository (p. 735)	O aplicativo sem servidor do portal do desenvolvedor do Amazon API Gateway já está disponível no AWS Serverless Application Repository (além do GitHub). Para obter mais informações, consulte Usar um portal do desenvolvedor para catalogar as APIs do API Gateway .	November 16, 2018
Suporte para AWS WAF (p. 735)	Suporte adicionado para AWS WAF (Firewall de aplicativos web). Para obter mais informações, consulte Controlar o acesso a uma API usando o AWS WAF .	November 5, 2018
Portal do desenvolvedor sem servidor (p. 735)	Agora o Amazon API Gateway fornece um portal do desenvolvedor totalmente personalizável como um aplicativo 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 .	October 29, 2018
Suporte para cabeçalhos de vários valores e parâmetros de string de consulta (p. 735)	Agora o Amazon API Gateway oferece suporte a vários cabeçalhos e parâmetros de string de consulta que possuem o mesmo nome. Para obter mais informações, consulte Suporte para cabeçalhos de vários valores e parâmetros de string de consulta .	October 4, 2018
Suporte ao OpenAPI (p. 735)	Agora o Amazon API Gateway oferece suporte ao OpenAPI 3.0 além do OpenAPI (Swagger) 2.0.	September 27, 2018
Documentação atualizada (p. 735)	Adição de um novo tópico: Como as políticas de recursos do Amazon API Gateway afetam o fluxo de trabalho de autorização .	September 27, 2018

Integração do AWS X-Ray ativa (p. 735)	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 .	September 6, 2018
Melhorias no armazenamento em cache (p. 735)	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 .	August 20, 2018
Limites de serviço revistos (p. 735)	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 mais informações, consulte Limites .	July 13, 2018
Substituição de parâmetros e cabeçalhos de solicitação e resposta de API (p. 735)	Supporte 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 .	July 12, 2018

Limitação de uso em nível de método para planos de uso (p. 735)	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 Controlar as solicitações de API para uma melhor produtividade .	July 11, 2018
Notificações de atualização do Guia do desenvolvedor do API Gateway agora disponível por meio de RSS (p. 735)	A versão HTML do Guia do desenvolvedor do API Gateway agora oferece suporte a um feed RSS de atualizações que estão documentadas nessa página Histórico da 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.	June 27, 2018

Atualizações anteriores

A tabela a seguir descreve alterações importantes em cada versão do Guia do desenvolvedor do API Gateway antes de 27 de junho de 2018.

Mudança	Descrição	Alterado em
APIs privadas	Adição de suporte para APIs privadas (p. 200) , que são expostas por meio de VPC endpoints de interface . O tráfego para as APIs privadas não deixa a rede da Amazon; ele é isolado da Internet pública.	14 de junho de 2018
Integrações e autorizadores do Lambda entre contas e autorizadores de grupo de usuários do Amazon Cognito entre contas	Use uma função do AWS Lambda de uma conta diferente da AWS como uma função de autorizador do Lambda ou como um back-end de integração da API. Ou use um grupo de usuários do Amazon Cognito como um autorizador. A outra conta pode estar em qualquer região em que o Amazon API Gateway estiver disponível. Para obter mais informações, consulte the section called “Configurar um autorizador do Lambda entre contas” (p. 412) , the section called “TUTORIAL: Criar uma API com integração de proxy do Lambda entre contas” (p. 34) e the	2 de abril de 2018

Mudança	Descrição	Alterado em
	section called “Configurar o autorizador do Amazon Cognito entre contas para uma API REST” (p. 422).	
Políticas de recursos para APIs	Use as políticas de recursos do API Gateway para permitir que os usuários de uma conta diferente da AWS 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 the section called “Usar políticas de recursos do API Gateway” (p. 362).	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 the section called “Configurar tags” (p. 534).	19 de dezembro de 2017
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 the section called “Habilitar compactação de carga” (p. 339).	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 the section called “Escolha de uma chave de API” (p. 452).	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 the section called “Use o Grupo de usuários do Cognito como um autorizador para uma API REST” (p. 414).	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 VpcLink . Para obter mais informações, consulte the section called “TUTORIAL: Criar uma API com integração privada” (p. 89) e the section called “Configurar integrações privadas” (p. 252).	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. Você pode definir uma porcentagem de tráfego do estagio para a versão canary e permitir a execução específica do canary e o acesso conectado em logs do CloudWatch Logs separados. Para obter mais informações, consulte the section called “Definir uma implantação da versão Canary” (p. 704).	28 de novembro de 2017
Registro de acesso em logs	Registre o acesso do cliente à sua API em logs com dados derivados de variáveis \$context (p. 306) em um formato de sua escolha. Para obter mais informações, consulte the section called “Configurar registro de API em logs do CloudWatch” (p. 536).	21 de novembro de 2017

Mudança	Descrição	Alterado em
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 the section called “Gerar o SDK do Ruby de uma API” (p. 550) e the section called “Usar um SDK Ruby gerado pelo API Gateway para uma API REST” (p. 573) .	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 the section called “Configurar uma API regional” (p. 197) e the section called “Configurar um nome de domínio personalizado regional para a API WebSocket ou REST” (p. 692) .	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 Usar autorizadores do Lambda do API Gateway (p. 396) .	15 de setembro de 2017
Como personalizar respostas de gateway	Personalize respostas de gateway geradas pelo API Gateway a solicitações de API que não conseguiram atingir o back-end 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 Configurar respostas do gateway para personalizar respostas de erro (p. 263) .	6 de junho de 2017
Mapeando 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 Lidar com erros personalizados do Lambda no API Gateway (p. 246) .	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 Controlar o fluxo de solicitações de API para uma melhor produtividade (p. 531) .	6 de junho de 2017

Mudança	Descrição	Alterado em
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 Ativar a validação de solicitação no API Gateway (p. 343) .	11 de abril de 2017
Integração 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 Configurar um nome de domínio personalizado para uma API no API Gateway (p. 676) .	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 Usar um SDK Java gerado pelo API Gateway para uma API REST (p. 566) .	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. Permita que o API Gateway lide com a autorização de usuários e a medição do uso. Para obter mais informações, consulte Vender suas APIs como SaaS (p. 673) .	1 de dezembro de 2016
Habilitando o suporte a documentação para a sua API	Adicione a documentação para entidades de API em recursos DocumentationPart no API Gateway. Associe um snapshot das instâncias de DocumentationPart da coleção com um estágio de API para criar um DocumentationVersion . Publique a documentação da API exportando uma versão da documentação para um arquivo externo, como um arquivo do Swagger. Para obter mais informações, consulte Documentação de uma API REST no API Gateway (p. 466) .	1 de dezembro de 2016
Autorizador personalizado atualizado	Agora, uma função Lambda de autorizador personalizado retorna o identificador principal do agente de 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 Saída de um autorizador do Lambda do Amazon API Gateway (p. 408) .	1 de dezembro de 2016
Oferecendo suporte a cargas binárias	Defina binaryMediaTypes na sua API para oferecer suporte a cargas binárias de uma solicitação ou resposta. Defina a propriedade contentHandling em um recurso Integration ou IntegrationResponse 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 Oferecer suporte a cargas úteis binárias no API Gateway (p. 316) .	17 de novembro de 2016

Mudança	Descrição	Alterado em
Habilitando uma integração de proxy com um back-end 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 com um back-end HTTP ou Lambda usando a integração de proxy HTTP ou Lambda, respectivamente. Para obter mais informações, consulte Configurar a integração de proxy com um recurso de proxy (p. 222) .	20 de setembro de 2016
Estendendo 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 Criação e utilização de planos de uso com chaves de API (p. 450) .	11 de agosto de 2016
Habilitando a autorização em nível de método com um pool de usuários no Amazon Cognito	Crie um grupo de usuários no Amazon Cognito e use-o como seu próprio provedor de identidade. 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 Controle o acesso à API REST usando o Grupos de usuários do Amazon Cognito como um autorizador (p. 414) .	28 de julho de 2016
Habilitando métricas e dimensões do Amazon CloudWatch no namespace AWS/ApiGateway.	Agora, as métricas do API Gateway estão padronizadas no namespace AWS/ApiGateway do CloudWatch. Você pode visualizá-las tanto no console do API Gateway quanto no console do Amazon CloudWatch. Para obter mais informações, consulte Amazon API Gateway Dimensões e métricas (p. 599) .	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 Revezar um certificado importado para o ACM (p. 690) .	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 TUTORIAL: Crie uma API REST importando um exemplo (p. 45) e TUTORIAL: Criar uma API com integração não proxy HTTP (p. 59) .	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, você pode criar uma nova API ou atualizar uma existente, carregando 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 Importar uma API REST no API Gateway (p. 356) .	5 de abril de 2016

Mudança	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 Referência de variáveis de registro de acesso em logs e modelo de mapeamento do API Gateway (p. 305) .	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 Descarregar o cache de estágios de API no API Gateway (p. 528) e Invalidate uma entrada de cache do API Gateway (p. 528) . Melhore a experiência do console para gerenciar o controle de fluxo de solicitações de API. Para obter mais informações, consulte Controlar o fluxo de solicitações de API para uma melhor produtividade (p. 531) .	25 de março de 2016
Habilitando e chamando 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 Usar autorizadores do Lambda do API Gateway (p. 396) .	11 de fevereiro de 2016
Importando e exportando 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 o recurso Export API do API Gateway. Para obter mais informações, consulte Importar uma API REST no API Gateway (p. 356) e Exportar uma API REST (p. 700) .	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 Referência de mapeamento de dados de solicitações e respostas de API do Amazon API Gateway (p. 301) .	18 de dezembro de 2015
Trabalhando 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 Configurar variáveis de estágio para a implantação de uma API REST (p. 540) .	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 Habilitar o CORS para um recurso da API REST (p. 423) .	3 de novembro de 2015

Mudança	Descrição	Alterado em
Como: usar a autenticação SSL de cliente	Use o Amazon API Gateway para gerar certificados SSL que você pode usar para autenticar chamadas para o seu backend HTTP. Para obter mais informações, consulte Gerar e configurar um certificado SSL para autenticação de back-end (p. 429) .	22 de setembro de 2015
Integração simulada de métodos	Saiba como integrar uma API por simulação com o Amazon API Gateway (p. 260) . Esse recurso permite que os desenvolvedores gerem respostas de API no API Gateway diretamente, sem necessidade de um back-end de integração final antecipado.	1 de setembro de 2015
Suporte a identidades do Amazon Cognito	O Amazon API Gateway expandiu o escopo da variável <code>\$context</code> de modo que ela agora retorna informações sobre identidade do Amazon Cognito quando 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 Referência de variáveis de registro de acesso em logs e modelo de mapeamento do API Gateway (p. 305) .	28 de agosto de 2015
Integração com o Swagger	Use a ferramenta de importação do Swagger no GitHub para importar as definições de API do Swagger para o Amazon API Gateway. Saiba mais sobre Extensões do API Gateway para OpenAPI (p. 606) 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 Referência de variáveis de registro de acesso em logs e modelo de mapeamento do API Gateway (p. 305) .	18 de julho de 2015
Lançamento público inicial	Este é o lançamento público inicial do Guia do desenvolvedor do API Gateway.	9 de julho de 2015

AWS Glossary

For the latest AWS terminology, see the [AWS Glossary](#) in the AWS General Reference.